1. Spark是什么?

Spark是大数据的调度，监控和分配引擎。它是一个快速通用的集群计算平台.Spark扩展了流行的MapReduce模型.Spark提供的主要功能之一就是能够在内存中运行计算 ， 但对于在磁盘上运行的复杂应用程序，系统也比MapReduce更有效。

2. 解释Spark的主要功能?

- 多语言
- 速度
- 多格式支持
- 延迟执行
- 实时计算
- Hadoop集成
- 机器学习

多语言：Spark提供Java，Scala，Python和R中的高级API .Spark代码可以用这四种语言中的任何一种编写。 它为Scala和Python提供了shell。 可以通过./bin/spark-shell进入Scala Shell和通过./bin/pyspark 进入Python shell

速度：Spark的运行速度比Hadoop MapReduce快100倍，适用于大规模数据处理。 Spark能够通过控制分区实现此速度。 它使用分区管理数据，这些分区有助于以最小的网络流量并行化处理分布式数据处

多种格式：Spark支持多种数据源，如Parquet，JSON，Hive和Cassandra。 Data Sources API提供了一种可插入的机制，用于通过Spark SQL访问结构化数据。 数据源不仅仅是简单的管道，可以转换数据并将其拉入Spark。

延迟执行：Apache Spark延迟执行，直到绝对必要。 这是促进其速度的关键因素之一。 对于transformations，Spark将它们添加到计算的DAG中，并且仅当驱动程序请求某些数据时，才会实际执行此DAG。

实时计算：Spark的计算是实时的，并且由于其内存计算而具有较少的延迟。 Spark专为大规模可扩展性而设计，Spark团队已经记录了运行具有数千个节点的生产集群的系统用户，并支持多种计算模型。
Hadoop集成：Apache Spark提供与Hadoop的兼容性。 Spark是Hadoop的MapReduce的潜在替代品，而Spark能够使用YARN在现有的Hadoop集群上运行资源调度。

机器学习：Spark的MLlib是机器学习组件，在大数据处理方面很方便。 它消除了使用多种工具的需求，一种用于处理，一种用于机器学习。 Spark为数据工程师和数据科学家提供了一个功能强大，统一的引擎，既快速又易于使用。

3. 我们如何在Spark中创建RDD?

Spark提供了两种创建RDD的方法：

1.通过在Driver程序中并行化集合。

2.这使用了SparkContext的'parallelize。

```
method val DataArray = Array(2,4,6,8,10)
val DataRDD = sc.parallelize(DataArray)
```

3.通过从外部存储器加载外部数据集，如HDFS，HBase，共享文件系统。

4. 如何在Apache Spark中定义分区?

顾名思义，分区是一个较小的逻辑分区，类似于MapReduce中的“split”。 它是大型分布式数据集的逻辑块。 分区是导出数据逻辑单元以加速处理过程的过程。 Spark使用分区来管理数据，这些分区有助于并行化分布式数据处理，只需最少的网络流量，Spark会尝试从靠近它的节点将数据读入RDD。 由于Spark通常访问分布式分区数据，为了优化转换操作，它创建了用于保存数据块的分区。

5. RDD支持哪些操作?

RDD（Resilient Distributed Dataset）是Spark中的主要逻辑数据单元。 RDD是分布式集合对象。 分布式意味着，每个RDD分为多个分区。 这些分区中的每一个都可以驻留在内存中或存储在群集中不同计算机的磁盘上。 RDD是不可变（只读）数据结构。 无法更改原始RDD，但可以将其转换为所需的不同RDD。

RDD支持两种类型的操作：转换和操作。

转换（Transformations）：转换从现有的RDD创建新的RDD，如map，reduceByKey和我们刚看到的过滤器（filter）。 转换按需执行。 这意味着它们是懒惰计算的。

操作（Actions）：操作返回RDD计算的最终结果。 Actions使用lineage graph触发执行以将数据加载到原始RDD中，执行所有中间转换并将最终结果返回到驱动程序或将其写入文件系统。

6. 你对Spark中的转换（Transformations）有什么了解?

Transformations是应用于RDD的函数，导致另一个RDD。 在动作发生之前它不会执行。 map（）和filter（）是转换的示例，前者应用在RDD的每个元素上传递给它的函数，并导致另一个RDD。filter（）通过从当前RDD中选择传递函数参数的元素来创建新的RDD。

```
val rawData=sc.textFile("path to/movies.txt")
val moviesData=rawData.map(x => x.split("\t"))
```

正如我们在这里看到的，rawData RDD被转换为moviesData RDD。 Transformations延迟执行

7. 如何在Spark中定义操作（Actions）？

Actions有助于将数据从RDD取到本地。Actions的执行是所有先前创建的transformation的结果。Actions使用 lineage graph触发执行以将数据加载到原始RDD中，执行所有中间转换并将最终结果返回到驱动程序或将其写入文件系统。

reduce() 是一个动作，它实现了一次又一次传递的函数，直到剩下一个值。 take() action将RDD中的所有值都带到本地节点。

```
moviesData.saveAsTextFile("MoviesData.txt")
```

正如我们在这里看到的，moviesData RDD被保存到名为MoviesData.txt的文本文件中。

8. SparkCore有什么功能?

Spark Core是大规模并行和分布式数据处理的基础引擎。 核心是分布式执行引擎，Java，Scala和Python API为分布式ETL应用程序开发提供了一个平台。 SparkCore执行各种重要功能，如内存管理，监视作业，容错，作业调度以及与存储系统的交互。 此外，在core上构建的其他库允许streaming，SQL和机器学习的各种工作负载。 它负责：

- 内存管理和故障恢复
- 在群集上调度，分发和监视作业
- 与存储系统交互

9. Spark 生态组件有哪些?

- Spark Core：用于大规模并行和分布式数据处理的基础引擎
- Spark Streaming：用于处理实时流数据
- Spark SQL：将关系处理与Spark的函数式编程API集成在一起
- GraphX：图形和图形并行计算
- MLlib：在Apache Spark中执行机器学习

10. 谈谈RDD、DataFrame、Dataset的区别和各自的优势?

共性：
1、RDD、DataFrame、Dataset全都是spark平台下的分布式弹性数据集，为处理超大型数据提供便利。
2、三者都有惰性机制，在进行创建、转换，如map方法时，不会立即执行，只有在遇到Action如foreach时，三者才会开始遍历运算。
3、三者都会根据spark的内存情况自动缓存运算，这样即使数据量很大，也不用担心会内存溢出。
4、三者都有partition的概念。

5、三者有许多共同的函数，如filter，排序等
6、在对DataFrame和Dataset进行操作许多操作都需要这个包进行支持：
import spark.implicits._
7、DataFrame和Dataset均可使用模式匹配获取各个字段的值和类型。
区别：
RDD:
1、RDD一般和spark mlib同时使用
2、RDD不支持sparksql操作
DataFrame:
1、与RDD和Dataset不同，DataFrame每一行的类型固定为Row，只有通过解析才能获取各个字段的值。
2、DataFrame与Dataset均支持sparksql的操作，比如select，groupby之类，还能注册临时表/视窗，进行sql语句操作。
3、DataFrame与Dataset支持一些特别方便的保存方式，比如保存成csv，可以带上表头，这样每一列的字段名一目了然。
Dataset:
这里主要对比Dataset和DataFrame，因为Dataset和DataFrame拥有完全相同的成员函数，区别只是每一行的数据类型不同。
DataFrame也可以叫Dataset[Row],每一行的类型是Row，不解析，每一行究竟有哪些字段，各个字段又是什么类型都无从得知，只能用上面提到的getAS方法或者共性中的第七条提到的模式匹配拿出特定字段。
而Dataset中，每一行是什么类型是不一定的，在自定义了case class之后可以很自由的获得每一行的信息。

11. 什么是RDD血统?

Spark不支持内存中的数据副本，因此，如果丢失了任何数据，则使用RDD血统进行重建。 RDD血统是一个重建丢失的数据分区的过程，因此 最好的是RDD记住如何从其他数据集构建。

12. 什么是Spark Driver?

Spark Driver是在机器的master上运行的程序，用于声明数据RDD上的转换（transformations）和操作（actions）。 简单来说，Spark中的驱动程序创建SparkContext，连接到给定的Spark Master。
该驱动程序还将RDD graph提供给Master，其中 standalone cluster 管理器运行。

13. Spark支持哪些文件系统?

Spark支持以下三个文件系统：
Hadoop分布式文件系统（HDFS）。
本地文件系统。
亚马逊S3

14. 什么是Spark Executor?

当SparkContext连接到集群管理器时，它会在集群中的节点上获取Executor。 executor是Spark进程，它运行计算并将数据存储在工作节点上。 SparkContext的最终任务被转移到executors以执行它们。

15. Spark中的集群的类型?

Standalone：设置群集的基本manager。
Apache Mesos：通用/常用的集群manager，也运行Hadoop MapReduce和其他应用程序。
YARN：负责Hadoop的资源管理。

16. 如何理解延迟调用?

Spark在数据运作方式很聪明。 当你告诉Spark对一个给定的数据集进行操作时，它会注意到指令并对其进行记录，这样它就不会忘记 - 但它什么也不做，除非要求最终结果。 在RDD上调用map（）之类的转换时，不会立即执行该操作。Transformations不会被执行，直到action操作时，执行操作 。 这有助于优化整体数据处理工作流程。

17. SchemaRDD在Spark RDD中你有什么理解?

SchemaRDD是一个RDD，由行对象（基本字符串或整数数组周围的包装器）组成，其中包含有关每列中数据类型的架构信息。

SchemaRDD旨在让开发人员在日常的代码调试和SparkSQL核心模块上进行单元测试时更加轻松。 这个想法可以归结为使用类似于关系数据库模式的形式描述来描述RDD内部的数据结构。 除了常见的RDD API提供的所有基本功能之外，SchemaRDD还提供了一些通过SparkSQL实现的简单的关系查询接口函数。

现在，它正式重命名为Spark最新的主干上的DataFrame API。

18. 你对worker节点有什么了解?

Worker节点是指可以在集群中运行应用程序代码的任何节点。 驱动程序必须侦听并接受来自其executors的传入连接，并且必须可以从worker进行网络寻址。
worker节点基本上是从节点。 主节点（Master）分配工作，worker节点实际执行分配的任务。
worker节点处理存储在节点上的数据并将资源报告给Master。 基于资源可用性，Master调度任务。

19. 说说使用Spark的一些缺点?

以下是使用Apache Spark的一些缺点：
由于与Hadoop和MapReduce相比，Spark利用了更多的存储空间，因此可能会出现某些问题。
开发人员在Spark中运行应用程序时需要小心。
work必须分布在多个节点上，而不是在单个节点上运行所有内容。
Spark的“内存”功能在成本效益高的大数据处理方面可能成为瓶颈。

20. 列出一些Spark在处理过程中胜过Hadoop的用例?

传感器数据处理：Apache Spark的“内存”计算在这里工作得最好，因为数据是从不同来源检索和组合的。
实时处理：Spark优先于Hadoop进行实时数据查询。 例如 股票市场分析，银行，医疗保健，电信等
流处理：对于处理日志和检测实时流中的欺诈警报，Apache Spark是最佳解决方案。
大数据处理：在处理中型和大型数据集时，Spark的运行速度比Hadoop快100倍。

21. 可以使用Spark访问和分析存储在Cassandra数据库中的数据吗?

是的，如果使用Spark Cassandra Connector，则可以。要将Spark连接到Cassandra集群，需要将Cassandra Connector添加到Spark项目中。 在设置中，Spark执行程序将与本地Cassandra节点通信，并仅查询本地数据。 它通过减少网络的使用来在Spark executor（处理数据）和Cassandra节点（数据存在的地方）之间发送数据，从而加快查询速度。

22. 在使用Spark时，如何最大限度地减少数据传输?

最大限度地减少数据传输并避免混乱有助于编写以快速可靠的方式运行的spark程序。 使用Apache Spark时，可以最大限度地减少数据传输的各种方式：

使用广播变量 - 广播变量可提高小型和大型RDD之间的连接效率。
使用累加器 - 累加器有助于在执行时并行更新变量值。

最常见的方法是避免操作ByKey，重新分区或任何其他触发shuffle的操作。

23. Spark技术栈有哪些组件，每个组件都有什么功能，适合什么应用场景?

1）Spark core：是其它组件的基础，spark的内核，主要包含：有向循环图、RDD、Lingage、Cache、broadcast等，并封装了底层通讯框架，是Spark的基础。
2）SparkStreaming是一个对实时数据流进行高通量、容错处理的流式处理系统，可以对多种数据源（如Kafka、Flume、Twitter、Zero和TCP 套接字）进行类似Map、Reduce和Join等复杂操作，将流式计算分解成一系列短小的批处理作业。
3）Spark sql：Shark是SparkSQL的前身，Spark SQL的一个重要特点是其能够统一处理关系表和RDD，使得开发人员可以轻松地使用SQL命令进行外部查询，同时进行更复杂的数据分析。
4）BlinkDB ： 是一个用于在海量数据上运行交互式 SQL 查询的大规模并行查询引擎，它允许用户通过权衡数据精度来提升查询响应时间，其数据的精度被控制在允许的误差范围内。
5）MLBase是Spark生态圈的一部分专注于机器学习，让机器学习的门槛更低，让一些可能并不了解机器学习的用户也能方便地使用MLbase。MLBase分为四部分：MLlib、MLI、ML Optimizer和

MLRuntime。
6）GraphX是Spark中用于图和图并行计算

24. Apache Spark中的各种持久性级别是什么?

Apache Spark会自动保留来自各种shuffle操作的中间数据，但是，通常建议用户在计划重用RDD时调用RDD上的persist（）方法。 Spark具有各种持久性级别，可将RDD存储在磁盘或内存中，或者作为两者的组合，具有不同的复制级别。

Spark中的各种存储/持久性级别是：
1.MEMORY_ONLY
使用未序列化的Java对象格式，将数据保存在内存中。如果内存不够存放所有的数据，则数据可能就不会进行持久化。那么下次对这个RDD执行算子操作时，那些没有被持久化的数据，需要从源头处重新计算一遍。这是默认的持久化策略，使用cache()方法时，实际就是使用的这种持久化策略。

2.MEMORY_AND_DISK
使用未序列化的Java对象格式，优先尝试将数据保存在内存中。如果内存不够存放所有的数据，会将数据写入磁盘文件中，下次对这个RDD执行算子时，持久化在磁盘文件中的数据会被读取出来使用。

3.MEMORY_ONLY_SER
基本含义同MEMORY_ONLY。唯一的区别是，会将RDD中的数据进行序列化，RDD的每个partition会被序列化成一个字节数组。这种方式更加节省内存，从而可以避免持久化的数据占用过多内存导致频繁GC。

4.MEMORY_AND_DISK_SER
基本含义同MEMORY_AND_DISK。唯一的区别是，会将RDD中的数据进行序列化，RDD的每个partition会被序列化成一个字节数组。这种方式更加节省内存，从而可以避免持久化的数据占用过多内存导致频繁GC。

5.DISK_ONLY
使用未序列化的Java对象格式，将数据全部写入磁盘文件中。

6.MEMORY_ONLY_2, MEMORY_AND_DISK_2, 等等
对于上述任意一种持久化策略，如果加上后缀_2，代表的是将每个持久化的数据，都复制一份副本，并将副本保存到其他节点上。这种基于副本的持久化机制主要用于进行容错。假如某个节点挂掉，节点的内存或磁盘中的持久化数据丢失了，那么后续对RDD计算时还可以使用该数据在其他节点上的副本。如果没有副本的话，就只能将这些数据从源头处重新计算一遍了。

25. cache后面能不能接其他算子,它是不是action操作?

答：cache可以接其他算子，但是接了算子之后，起不到缓存应有的效果，因为会重新触发 cache。
cache不是action操作

26. reduceByKey是不是action?

答：不是，很多人都会以为是action，reduce rdd是action

27. 数据本地性是在哪个环节确定的?

具体的task运行在那他机器上，dag划分stage的时候确定的

28. RDD的弹性表现在哪几点?

1）自动的进行内存和磁盘的存储切换；
2）基于Lingage的高效容错；
3）task如果失败会自动进行特定次数的重试；
4）stage如果失败会自动进行特定次数的重试，而且只会计算失败的分片；
5）checkpoint和persist，数据计算之后持久化缓存
6）数据调度弹性，DAG TASK调度和资源无关
7）数据分片的高度弹性

29. 常规的容错方式有哪几种类型?

1）.数据检查点,会发生拷贝，浪费资源
2）.记录数据的更新，每次更新都会记录下来，比较复杂且比较消耗性能

30. RDD通过Linage（记录数据更新）的方式为何很高效?

1）lazy记录了数据的来源，RDD是不可变的，且是lazy级别的，且rDD 之间构成了链条，lazy是弹性的基石。由于RDD不可变，所以每次操作就 产生新的rdd，不存在全局修改的问题，控制难度下降，所有有计算链条 将复杂计算链条存储下来，计算的时候从后往前回溯 900步是上一个stage的结束，要么就checkpoint
2）记录原数据，是每次修改都记录，代价很大 如果修改一个集合，代价就很小，官方说rdd是 粗粒度的操作，是为了效率，为了简化，每次都是 操作数据集合，写或者修改操作，都是基于集合的 rdd的写操作是粗粒度的，rdd的读操作既可以是粗粒度的 也可以是细粒度，读可以读其中的一条条的记录。
3）简化复杂度，是高效率的一方面，写的粗粒度限制了使用场景 如网络爬虫，现实世界中，大多数写是粗粒度的场景

31. RDD有哪些缺陷?

1）不支持细粒度的写和更新操作（如网络爬虫），spark写数据是粗粒度的 所谓粗粒度，就是批量写入数据，为了提高效率。但是读数据是细粒度的也就是 说可以一条条的读
2）不支持增量迭代计算，Flink支持

32. 说一说Spark程序编写的一般步骤?

答：初始化，资源，数据源，并行化，rdd转化，action算子打印输出结果或者也可以存至相应的数据存储介质。

33. Spark有哪两种算子?

答：Transformation（转化）算子和Action（执行）算子。

34. Spark提交你的jar包时所用的命令是什么?

答：spark-submit

35. Spark有哪些聚合类的算子,我们应该尽量避免什么类型的算子?

答：在我们的开发过程中，能避免则尽可能避免使用reduceByKey、join、distinct、repartition 等会进行shuffle的算子，尽量使用map类的非shuffle算子。这样的话，没有shuffle操作或者仅有 较少shuffle操作的Spark作业，可以大大减少性能开销。

36. 你所理解的Spark的shuffle过程?

答：从下面三点去展开
1）shuffle过程的划分
2）shuffle的中间结果如何存储
3）shuffle的数据如何拉取过来
可以参考这篇博文：http://www.cnblogs.com/jxhd1/p/6528540.html

37. 你如何从Kafka中获取数据?

1)基于Receiver的方式 这种方式使用Receiver来获取数据。Receiver是使用Kafka的高层次Consumer API来实现的。receiver从Kafka中获取 的数据都是存储在Spark Executor的内存中的，然后Spark Streaming启动的job会去处理那些数据。
2)基于Direct的方式 这种新的不基于Receiver的直接方式，是在Spark 1.3中引入的，从而能够确保更加健壮的机制。替代掉使用Receiver来 接收数据后，这种方式会周期性地查询Kafka，来获得每个topic+partition的最新的offset，从而定义每个batch的 offset的范围。当处理数据的job启动时，就会使用Kafka的简单consumer api来获取Kafka指定offset范围的数据。

38. 对于Spark中的数据倾斜问题你有什么好的方案?

1）前提是定位数据倾斜，是OOM了，还是任务执行缓慢，看日志，看WebUI
2)解决方法，有多个方面
① 避免不必要的shuffle，如使用广播小表的方式，将reduce-side-join提升为map-side-join
② 分拆发生数据倾斜的记录，分成几个部分进行，然后合并join后的结果
③ 改变并行度，可能并行度太少了，导致个别task数据压力大 •两阶段聚合，先局部聚合，再全局聚合
•自定义paritioner，分散key的分布，使其更加均匀。
详细解决方案参考博文：《Spark数据倾斜优化方法》。

39. RDD创建有哪几种方式?

1).使用程序中的集合创建rdd
2).使用本地文件系统创建rdd
3).使用hdfs创建rdd，
4).基于数据库db创建rdd
5).基于Nosql创建rdd，如hbase
6).基于s3创建rdd，
7).基于数据流，如socket创建rdd。
如果只回答了前面三种，是不够的，只能说明你的水平还是入门级的，实践过程中有很多种创建方式。

40. Spark并行度怎么设置比较合适

答：spark并行度，每个core承载2~4个partition,如，32个core，那么64~128之间的并行度，也就是设置64~128个partion，并行读和数据规模无关，只和内存使用量和cpu使用时间有关。

41. Spark中数据的位置是被谁管理的?

答：每个数据分片都对应具体物理位置，数据的位置是被blockManager，无论 数据是在磁盘，内存还是tacyan，都是由blockManager管理

42. Spark的数据本地性有哪几种?

答：Spark中的数据本地性有三种：
a. PROCESS_LOCAL是指读取缓存在本地节点的数据
b. NODE_LOCAL是指读取本地节点硬盘数据
c. ANY是指读取非本地节点数据通常读取数据PROCESS_LOCAL>NODE_LOCAL>ANY，尽量使数据以PROCESS_LOCAL或 NODE_LOCAL方式读取。其中PROCESS_LOCAL还和cache有关，如果RDD经常用的话将该 RDD cache到内存中，注意，由于cache是lazy的，所以必须通过一个action的触发，才能真正的 将该RDD cache到内存中。

43. rdd有几种操作类型?

1）transformation，rdd由一种转为另一种rdd
2）action，
3）cronroller，crontroller是控制算子,cache,persist，对性能和效率的有很好的支持 三种类型，不要回答只有2种操作。

44. Spark如何处理不能被序列化的对象?

将不能序列化的内容封装成object

45. collect功能是什么，其底层是怎么实现的?

答：driver通过collect把集群中各个节点的内容收集过来汇总成结果，collect返回结果是Array类 型的，collect把各个节点上的数据抓过来，抓过来数据是Array型，collect对Array抓过来的结果 进行合并，合并后Array中只有一个元素，是tuple类型（KV类型的）的。

46. Spark程序执行，有时候默认为什么会产生很多task，怎么修改默认task执行个数?

答：1)因为输入数据有很多task，尤其是有很多小文件的时候，有多少个输入 block就会有多少个task启动；
2)spark中有partition的概念，每个partition都会对应一个task， task越多，在处理大规模数据的时候，就会越有效率。不过task并不是越多越好，如果平时测试， 或者数据量没有那么大，则没有必要task数量太多。
3)参数可以通过spark_home/conf/sparkdefault.conf配置文件设置: spark.sql.shuffle.partitions 50、spark.default.parallelism 10 第一个是针对spark sql的task数量 第二个是非spark sql程序设置生效。

47. 为什么Spark Application在没有获得足够的资源，job就开始执行了，可能会导致什么什么 问题发生?

答：会导致执行该job时候集群资源不足，导致执行job结束也没有分配足够的资源，分配了部分Executor，该job就开始执行task，应该是task的调度线程和Executor资源申请是异步的；如果想等待申请完所有的资源再执行job的：需要将 spark.scheduler.maxRegisteredResourcesWaitingTime设置的很大；spark.scheduler.minRegisteredResourcesRatio 设置为1，但是应该结合实际考虑 否则很容易出现长时间分配不到资源，job一直不能运行的情况。

48. map与flatMap的区别

map：对RDD每个元素转换，文件中的每一行数据返回一个数组对象 flatMap：对RDD每个元素转换，然后再扁平化 将所有的对象合并为一个对象，文件中的所有行数据仅返回一个数组 对象，会抛弃值为null的值。

49. 列举你常用的action?

collect，reduce,take,count,saveAsTextFile等。

50. Spark为什么要持久化，一般什么场景下要进行persist操作？ 为什么要进行持久化?

spark所有复杂一点的算法都会有persist身影,spark默认数据放在内存，spark很多内容都是放在内存的，非常适合高速迭代，1000个步骤，只有第一个输入数据，中间不产生临时数据，但分布式系统风险很高，所以容易出错，就要容错，rdd出错或者分片可以根据血统算出来，如果没有对父rdd进行persist 或者cache的化，就需要重头做。
以下场景会使用persist
1）某个步骤计算非常耗时，需要进行persist持久化
2）计算链条非常长，重新恢复要算很多步骤，很好使，persist
3）checkpoint所在的rdd要持久化persist， lazy级别，框架发现有checnkpoint，checkpoint时单独触发一个job，需要重算一遍， checkpoint前 要持久化，写个rdd.cache或者rdd.persist，将结果保存起来，再写checkpoint操作，这样执行 起来会非常快，不需要重新计算rdd链条了。checkpoint之前一定会进行persist。

4）shuffle之后为什么要persist，shuffle要进性网络传输，风险很大，数据丢失重来，恢复代价 很大
5）shuffle之前进行persist，框架默认将数据持久化到磁盘，这个是框架自动做的。

51. 为什么要进行序列化

序列化可以减少数据的体积，减少存储空间，高效存储和传输数据，不好的是使用的时候要反序 列化，非常消耗CPU。

52. 介绍一下cogroup rdd实现原理，你在什么场景下用过这个rdd?

答：cogroup的函数实现:
这个实现根据两个要进行合并的两个RDD操作,生成一个 CoGroupedRDD的实例,这个RDD的返回结果是把相同的key中两个RDD分别进行合并操作,最后 返回的RDD的value是一个Pair的实例,这个实例包含两个Iterable的值,第一个值表示的是RDD1中 相同KEY的值,第二个值表示的是RDD2中相同key的值。由于做cogroup的操作,需要通过 partitioner进行重新分区的操作,因此,执行这个流程时,需要执行一次shuffle的操作(如果要进行合 并的两个RDD的都已经是shuffle后的rdd,同时他们对应的partitioner相同时,就不需要执行 shuffle)。
场景：表关联查询

53. 下面这段代码输出结果是什么?

```
def joinRdd(sc:SparkContext) {
val name= Array(
Tuple2(1,"spark"),
Tuple2(2,"tachyon"),
Tuple2(3,"hadoop")
)
val score= Array(
Tuple2(1,100),
Tuple2(2,90),
Tuple2(3,80)
)
val namerdd=sc.parallelize(name);
val scorerdd=sc.parallelize(score);
val result = namerdd.join(scorerdd);        result .collect.foreach(println);

}
```

答案：(1,(Spark,100))
(2,(tachyon,90))
(3,(hadoop,80))

54. Executor之间如何共享数据?

答：基于hdfs或者基于tachyon

55. Spark累加器有哪些特点?

1）累加器在全局唯一的，只增不减，记录全局集群的唯一状态
2）在exe中修改它，在driver读取
3）executor级别共享的，广播变量是task级别的共享 两个application不可以共享累加器，但是同一个app不同的job可以共享。

56. spark hashParitioner的弊端是什么?

答:HashPartitioner分区的原理很简单，对于给定的key，计算其hashCode，并除于分区的个数 取余，如果余数小于0，则用余数+分区的个数，最后返回的值就是这个key所属的分区ID；弊端 是数据不均匀，容易导致数据倾斜，极端情况下某几个分区会拥有rdd的所有数据。

57. RangePartitioner分区的原理?

答:RangePartitioner分区则尽量保证每个分区中数据量的均匀，而且分区与分区之间是有序的， 也就是说一个分区中的元素肯定都是比另一个分区内的元素小或者大；但是分区内的元素是不能 保证顺序的。简单的说就是将一定范围内的数映射到某一个分区内。其原理是水塘抽样。可以参考这篇博文 https://www.iteblog.com/archives/1522.html

58. Spark应用程序的执行过程是什么?

1).构建Spark Application的运行环境（启动SparkContext），SparkContext向资源管理器（可以是Standalone、Mesos或YARN）注册并申请运行Executor资源；
2).资源管理器分配Executor资源并启动StandaloneExecutorBackend，Executor运行情况将随 着心跳发送到资源管理器上；
3).SparkContext构建成DAG图，将DAG图分解成Stage，并把Taskset发送给Task Scheduler。Executor向SparkContext申请Task，Task Scheduler将Task发放给Executor运行同时 SparkContext将应用程序代码发放给Executor。
4).Task在Executor上运行，运行完毕释放所有资源。

59. 如何理解Standalone模式下，Spark资源分配是粗粒度的?

答：spark默认情况下资源分配是粗粒度的，也就是说程序在提交时就分配好资源，后面执行的时 候 使用分配好的资源，除非资源出现了故障才会重新分配。比如Spark shell启动，已提交，一注册，哪怕没有任务，worker都会分配资源给executor。

60. Spark如何自定义partitioner分区器?

答：1）spark默认实现了HashPartitioner和RangePartitioner两种分区策略，我们也可以自己扩 展分区策略，自定义分区器的时候继承org.apache.spark.Partitioner类，实现类中的三个方法 def numPartitions: Int：这个方法需要返回你想要创建分区的个数； def getPartition(key: Any): Int：这个函数需要对输入的key做计算，然后返回该key的分区ID， 范围一定是0到numPartitions-1；equals()：这个是Java标准的判断相等的函数，之所以要求用户实现这个函数是因为Spark内部会 比较两个RDD的分区是否一样。
2）使用，调用parttionBy方法中传入自定义分区对象
参考：http://blog.csdn.net/high2011/article/details/68491115

61. spark中task有几种类型?

答：2种类型：1）result task类型，最后一个task，2是shuffleMapTask类型，除了最后一个 task都是

62. union操作是产生宽依赖还是窄依赖?

答：窄依赖

63. rangePartioner分区器特点?

答：rangePartioner尽量保证每个分区中数据量的均匀，而且分区与分区之间是有序的，一个分 区中的元素肯定都是比另一个分区内的元素小或者大；但是分区内的元素是不能保证顺序的。简 单的说就是将一定范围内的数映射到某一个分区内。
RangePartitioner作用：将一定范围内的数 映射到某一个分区内，在实现中，分界的算法尤为重要。算法对应的函数是rangeBounds。

64. 什么是二次排序，你是如何用spark实现二次排序的？（互联网公司常面）

答：就是考虑2个维度的排序，key相同的情况下如何排序，参考博文： http://blog.csdn.net/sundujing/article/details/51399606

65. 如何使用Spark解决TopN问题？（互联网公司常见的面试题）

答：参考博文：http://www.cnblogs.com/yurunmiao/p/4898672.html

66. 如何使用Spark解决分组排序问题？（互联网公司常面）

组织数据形式：
aa 11
bb 11
cc 34
aa 22
bb 67
cc 29
aa 36
bb 33
cc 30
aa 42
bb 44
cc 49
需求：
1、对上述数据按key值进行分组

2、对分组后的值进行排序
3、截取分组后值得top 3位以key-value形式返回结果

答案如下 ----------

```
Val groupTopNRdd =sc.textFile(
"hdfs://db02:8020/user/hadoop/groupsorttop/groupsorttop.data"
)
groupTopNRdd.map(.split(" "))
.map(x => (x(0),x(1)))
.groupByKey()
.map( x => {
val xx = x.1
val yy = x._2
(xx,yy.toList.sorted.reverse.take(3))
} ).collect
```

67. 窄依赖父RDD的partition和子RDD的parition是不是都是一对一的关系?

答：不一定，除了一对一的窄依赖，还包含一对固定个数的窄依赖（就是对父RDD的依赖的 Partition 的数量不会随着RDD数量规模的改变而改变），比如join操作的每个partiion仅仅和已知 的partition进行join，这个join操作是窄依赖，依赖固定数量的父rdd，因为是确定的partition关系。

68. Hadoop中，Mapreduce操作的mapper和reducer阶段相当于spark中的哪几个算子?

答：相当于spark中的map算子和reduceByKey算子，当然还是有点区别的,MR会自动进行排序的，spark要看你用的是什么partitioner。

69. 什么是shuffle，以及为什么需要shuffle?

shuffle中文翻译为洗牌，需要shuffle的原因是：某种具有共同特征的数据汇聚到一个计算节点上进行计算。

70. 不需要排序的hash shuffle是否一定比需要排序的sort shuffle速度快?

答：不一定！！当数据规模小，Hash shuffle快于Sorted Shuffle数据规模大的时候；当数据量 大，sorted Shuffle会比Hash shuffle快很多，因为数量大的有很多小文件，不均匀，甚至出现数 据倾斜，消耗内存大，1.x之前spark使用hash，适合处理中小规模，1.x之后，增加了Sorted shuffle，Spark更能胜任大规模处理了。

71. Spark中的HashShufle的有哪些不足?

答：1）shuffle产生海量的小文件在磁盘上，此时会产生大量耗时的、低效的IO操作；
2）.容易导 致内存不够用，由于内存需要保存海量的文件操作句柄和临时缓存信息，如果数据处理规模比较 大的化，容易出现OOM；
3）容易出现数据倾斜，导致OOM。

72. conslidate是如何优化Hash shuffle时在map端产生的小文件?

答：1）conslidate为了解决Hash Shuffle同时打开过多文件导致Writer handler内存使用过大以 及产生过多文件导致大量的随机读写带来的低效磁盘IO；2）conslidate根据CPU的个数来决定每 个task shuffle map端产生多少个文件，假设原来有10个task，100个reduce，每个CPU有10个 CPU 那么使用hash shuffle会产生10*100=1000个文件，conslidate产生10*10=100个文件 备注：conslidate部分减少了文件和文件句柄，并行读很高的情况下（task很多时）还是会很多文件。

73. Sort-basesd shuffle产生多少个临时文件?

答：2*Map阶段所有的task数量，Mapper阶段中并行的Partition的总数量，其实就是Mapper端task。

74. Sort-based shuffle的缺陷?

1）如果mapper中task的数量过大，依旧会产生很多小文件，此时在shuffle传递数据的过程中 reducer段，reduce会需要同时大量的记录进行反序列化，导致大量的内存消耗和GC的巨大负 担，造成系统缓慢甚至崩溃
2）如果需要在分片内也进行排序，此时需要进行mapper段和reducer段的两次排序。

75. Spark shell启动时会启动derby?

答： spark shell启动会启动spark sql，spark sql默认使用derby保存元数据，但是尽量不要用 derby，它是单实例，不利于开发。会在本地生成一个文件metastore_db,如果启动报错，就把那 个文件给删了，derby数据库是单实例，不能支持多个用户同时操作，尽量避免使用。

76. spark.default.parallelism这个参数有什么意义，实际生产中如何设置?

答：1）参数用于设置每个stage的默认task数量。这个参数极为重要，如果不设置可能会直接影 响你的Spark作业性能；
3）很多人都不会设置这个参数，会使得集群非常低效，你的cpu，内存 再多，如果task始终为1，那也是浪费，spark官网建议task个数为CPU的核数*executor的个数的 2~3倍。

77. 提交任务时，如何指定Spark Application的运行模式?

1）cluster模式：./spark-submit --class xx.xx.xx --master yarn --deploy-mode cluster xx.jar
2）client模式:./spark-submit --class xx.xx.xx --master yarn --deploy-mode client xx.jar

78. 不启动Spark集群Master和work服务，可不可以运行Spark程序?

答：可以，只要资源管理器第三方管理就可以，如由yarn管理，spark集群不启动也可以使用spark；spark集群 启动的是work和master，这个其实就是资源管理框架，yarn中的resourceManager相当于master， NodeManager相当于worker，做计算是Executor，和spark集群的work和manager可以没关系，归根接底还 是JVM的运行，只要所在的JVM上安装了spark就可以。

79. Spark中的4040端口由什么功能?

答：收集Spark作业运行的信息。

80. spark on yarn Cluster 模式下，ApplicationMaster和driver是在同一个进程么?

答：是,driver 位于ApplicationMaster进程中。该进程负责申请资源，还负责监控程序、资源的动态情况。

81. 如何使用命令查看application运行的日志信息?

答：yarn logs -applicationId

82. 谈谈你对container的理解?

1）Container作为资源分配和调度的基本单位，其中封装了的资源如内存，CPU，磁盘，网络带宽等。目前 yarn仅仅封装内存和CPU。
2)Container由ApplicationMaster向ResourceManager申请的，由ResouceManager中的资源调度器异步分 配给ApplicationMaster。
3) Container的运行是由ApplicationMaster向资源所在的NodeManager发起的，Container运行时需提供内 部执行的任务命令。

83. Executor启动时，资源通过哪几个参数指定?

1)num-executors是executor的数量
2)executor-memory 是每个executor使用的内存
3)executor-cores 是每个executor分配的CPU

84. Mapreduce的执行过程?

阶段1：input/map/partition/sort/spill
阶段2：mapper端merge
阶段3：reducer端merge/reduce/output 详细过程参考这个http://www.cnblogs.com/hipercomer/p/4516581.html

85. 一个task的map数量由谁来决定?

一般情况下，在输入源是文件的时候，一个task的map数量由splitSize来决定的，那么splitSize是由以下几个来决定的：
goalSize = totalSize / mapred.map.tasks
inSize = max {mapred.min.split.size, minSplitSize}
splitSize = max (minSize, min(goalSize, dfs.block.size))
一个task的reduce数量，由partition决定。

86. Combiner 和partition的作用

combine分为map端和reduce端，作用是把同一个key的键值对合并在一起，可以自定义的。combine函数把 一个map函数产生的<key,value>对（多个key,value）合并成一个新<key2,value2>.将新的<key2,value2>作 为输入到reduce函数中这个value2亦可称之为values，因为有多个。这个合并的目的是为了减少网络传输。 partition是分割map每个节点的结果，按照key分别映射给不同的reduce，也是可以自定义的。这里其实可以 理解归类。我们对于错综复杂的数据归类。比如在动物园里有牛羊鸡鸭鹅，他们都是混在一起的，但是到了晚上 他们就各自牛回牛棚，羊回羊圈，鸡回鸡窝。partition的作用就是把这些数据归类。只不过在写程序的时候， mapreduce使用哈希HashPartitioner帮我们归类了。这个我们也可以自定义。shuffle就是map和reduce之间 的过程，包含了两端的combine和partition。Map的结果，会通过partition分发到Reducer上，Reducer做完 Reduce操作后，通OutputFormat，进行输出shuffle阶段的主要函数是fetchOutputs(),这个函数的功能就是将 map阶段的输出，copy到reduce 节点本地。

87. 哪些算子操作涉及到shuffle?

sortByKey、groupByKey、reduceByKey、countByKey、join、cogroup等聚合操作。
更多shuffle算子请参考：
https://blog.csdn.net/chixushuchu/article/details/86148575

88. 什么时候join不发生shuffle?

在大数据处理场景中，多表Join是非常常见的一类运算。为了便于求解，通常会将多表join问题转为多个两表连接问题。两表Join的实现算法非常多，一般我们会根据两表的数据特点选取不同的join算法，其中，最常用的两个算法是map-side join和reduce-side join。map-side join也就是join不产生Shuffle。
Map-side Join使用场景是一个大表和一个小表的连接操作，其中，“小表”是指文件足够小，可以加载到内存中。该算法可以将join算子执行在Map端，无需经历Shuffle和reduce等阶段，因此效率非常高。
Join是否发生shuffle详解参考：https://blog.csdn.net/weixin_38653290/article/details/83959059

89. driver的功能是什么?

1）一个Spark作业运行时包括一个Driver进程，也是作业的主进程，具有main函数，并且有SparkContext的实例，是程序的人口点；
2）功能：负责向集群申请资源，向master注册信息，负责了作业的调度，负责作业的解析、生成Stage并调度Task到Executor上。包括 DAGScheduler，TaskScheduler。

90. Spark中Work的主要工作是什么?

答：主要功能：管理当前节点内存，CPU的使用状况，接收master分配过来的资源指令，通过ExecutorRunner启动程序分配任务，worker就类似于包工头，管理分配新进程，做计算的服务， 相当于process服务。需要注意的是：1）worker会不会汇报当前信息给master，worker心跳给 master主要只有workid，它不会发送资源信息以心跳的方式给mater，master分配的时候就知道 work，只有出现故障的时候才会发送资源。
2）worker不会运行代码，具体运行的是Executor是 可以运行具体appliaction写的业务逻辑代码，操作代码的节点，它不会运行程序的代码的。

91. Spark为什么比mapreduce快?

答：1）基于内存计算，减少低效的磁盘交互；
2）高效的调度算法，基于DAG；
3)容错机制 Linage，精华部分就是DAG和Lingae

92. 简单说一下hadoop和spark的shuffle相同和差异?

答：
1）从 high-level 的角度来看，两者并没有大的差别。 都是将 mapper（Spark 里是 ShuffleMapTask）的输出进行 partition，不同的 partition 送到不同的 reducer（Spark 里 reducer 可能是下一个 stage 里的 ShuffleMapTask，也可能是 ResultTask）。Reducer 以内存 作缓冲区，边 shuffle 边 aggregate 数据，等到数据 aggregate 好以后进行 reduce() （Spark 里可能是后续的一系列操作）。
2）从 low-level 的角度来看，两者差别不小。 Hadoop MapReduce 是 sort-based，进入 combine() 和 reduce() 的 records 必须先 sort。这样的好处在于 combine/reduce() 可以处理 大规模的数据，因为其输入数据可以通过外排得到（mapper 对每段数据先做排序，reducer 的 shuffle 对排好序的每段数据做归并）。目前的 Spark 默认选择的是 hash-based，通常使用 HashMap 来对 shuffle 来的数据进行 aggregate，不会对数据进行提前排序。如果用户需要经过 排序的数据，那么需要自己调用类似 sortByKey() 的操作；如果你是Spark 1.1的用户，可以将 spark.shuffle.manager设置为sort，则会对数据进行排序。在Spark 1.2中，sort将作为默认的 Shuffle实现。
3）从实现角度来看，两者也有不少差别。 Hadoop MapReduce 将处理流程划分出明显的几个 阶段：map(), spill, merge, shuffle, sort, reduce() 等。每个阶段各司其职，可以按照过程式的编 程思想来逐一实现每个阶段的功能。在 Spark 中，没有这样功能明确的阶段，只有不同的 stage 和一系列的 transformation()，所以 spill, merge, aggregate 等操作需要蕴含在 transformation() 中。 如果我们将 map 端划分数据、持久化数据的过程称为 shuffle write，而将 reducer 读入数据、 aggregate 数据的过程称为 shuffle read。那么在 Spark 中，问题就变为怎么在 job 的逻辑或者 物理执行图中加入 shuffle write 和 shuffle read 的处理逻辑？以及两个处理逻辑应该怎么高效实 现？ Shuffle write由于不要求数据有序，shuffle write 的任务很简单：将数据 partition 好，并持久 化。之所以要持久化，一方面是要减少内存存储空间压力，另一方面也是为了 fault-tolerance。

93. RDD机制?

答：rdd分布式弹性数据集，简单的理解成一种数据结构，是spark框架上的通用货币。 所有算子都是基于rdd来执行的，不同的场景会有不同的rdd实现类，但是都可以进行互相转换。 rdd执行过程中会形成dag图，然后形成lineage保证容错性等。 从物理的角度来看rdd存储的是 block和node之间的映射。

94. spark有哪些组件?

答：主要有如下组件：
1）master：管理集群和节点，不参与计算。
2）worker：计算节点，进程本身不参与计算，和master汇报。
3）Driver：运行程序的main方法，创建spark context对象。
4）spark context：控制整个application的生命周期，包括dagsheduler和task scheduler等组 件。
5）client：用户提交程序的入口。

95. spark工作机制?

答：用户在client端提交作业后，会由Driver运行main方法并创建spark context上下文。 执行add算子，形成dag图输入dagscheduler，按照add之间的依赖关系划分stage输入task scheduler。 task scheduler会将stage划分为task set分发到各个节点的executor中执行。

96. spark-submit的时候如何引入外部jar包?

方法一：spark-submit –jars 根据spark官网，在提交任务的时候指定–jars，用逗号分开。这样做的缺点是每次都要指定jar 包，如果jar包少的话可以这么做，但是如果多的话会很麻烦。 命令：spark-submit --master yarn-client --jars ***.jar,***.jar
方法二：extraClassPath 提交时在spark-default中设定参数，将所有需要的jar包考到一个文件里，然后在参数中指定该目 录就可以了，较上一个方便很多：
spark.executor.extraClassPath=/home/hadoop/wzq_workspace/lib/*
spark.driver.extraClassPath=/home/hadoop/wzq_workspace/lib/*
需要注意的是,你要在所有可能运行spark任务的机器上保证该目录存在，并且将jar包考到所有机 器上。这样做的好处是提交代码的时候不用再写一长串jar了，缺点是要把所有的jar包都拷一遍。

97. cache和pesist的区别?

答：
1）cache和persist都是用于将一个RDD进行缓存的，这样在之后使用的过程中就不需要重新 计算了，可以大大节省程序运行时间；
2） cache只有一个默认的缓存级别MEMORY_ONLY ， cache调用了persist，而persist可以根据情况设置其它的缓存级别；
3）executor执行的时候，默 认60%做cache，40%做task操作，persist最根本的函数，最底层的函数。

98. Spark 的四大组件下面哪个不是 (D)

A. Spark Streaming B. Mlib C Graphx D.Spark R

99. 下面哪个端口不是 spark 自带服务的端口 (C)

A.8080 B.4040 C.8090 D.18080
备注：8080：spark集群web ui端口，4040：sparkjob监控端口，18080：jobhistory端口

100. spark 1.4 版本的最大变化 (B)

A spark sql Release 版本 B .引入 Spark R C DataFrame D.支持动态资源分配

101. Spark Job 默认的调度模式 (A)

A FIFO B FAIR C 无 D 运行时指定

102. 哪个不是本地模式运行的个条件 (D)

A spark.localExecution.enabled=true B 显式指定本地运行 C finalStage 无父 Stage D partition默认值

103. 下面哪个不是 RDD 的特点 (C)

A. 可分区 B 可序列化 C 可修改 D 可持久化

104. 关于广播变量，下面哪个是错误的 (D)

A 任何函数调用 B 是只读的 C 存储在各个节点 D 存储在磁盘或 HDFS

105. 关于累加器，下面哪个是错误的 (D)

A 支持加法 B 支持数值类型 C 可并行 D 不支持自定义类型
Spark 支持的分布式部署方式中哪个是错误的 (D)
A standalone B spark on mesos C spark on YARN D Spark on local

106. 10.Stage 的 Task 的数量由什么决定 (A)

A Partition B Job C Stage D TaskScheduler

107. 下面哪个操作是窄依赖 (B)

A join B filter C group D sort

108. 下面哪个操作肯定是宽依赖 (C)

A map B flatMap C reduceByKey D sample

109. spark 的 master 和 worker 通过什么方式进行通信的? （D ）

A http B nio C netty D Akka

110. 默认的存储级别 (A)

A MEMORY_ONLY B MEMORY_ONLY_SER
C MEMORY_AND_DISK D MEMORY_AND_DISK_SER

111. spark.deploy.recoveryMode 不支持那种 (D)

A.ZooKeeper B. FileSystem D NONE D Hadoop

112. 下列哪个不是 RDD 的缓存方法 (C)

A persist() B Cache() C Memory()

113. Task 运行在下来哪里个选项中 Executor 上的工作单元 (C)

A Driver program B. spark master C.worker node D Cluster manager

114. hive 的元数据存储在 derby 和 MySQL 中有什么区别 (B)

A.没区别 B.多会话 C.支持网络环境 D数据库的区别

115. DataFrame 和 RDD 最大的区别 (B)

A. 科学统计支持 B.多了 schema
B. 存储方式不一样 D.外部数据源支持

116. Master 的 ElectedLeader 事件后做了哪些操作 (D)

A. 通知 driver B.通知 worker C.注册 application D.直接 ALIVE

117. 简要描述 Spark 写数据的流程?

1）RDD 调用 compute 方法，进行指定分区的写入
2）CacheManager 中调用 BlockManager 判断数据是否已经写入，如果未写，则写入
3）BlockManager 中数据与其他节点同步
4）BlockManager 根据存储级别写入指定的存储层
5）BlockManager 向主节点汇报存储状态中

118. Spark 中 Lineage 的基本原理

这里应该是问你 Spark 的容错机制的原理：
1）Lineage（又称为 RDD 运算图或 RDD 依赖关系图）是 RDD 所有父 RDD 的 graph（图）。它 是在 RDD 上执行 transformations 函数并创建 logical execution plan（逻辑执行计划） 的结果，是 RDD 的逻辑执行计划，记录了 RDD 之间的依赖关系。
2)使用 Lineage 实现 spark 的容错，本质上类似于数据库中重做日志，是容错机制的一 种方式，不过这个重做日志粒度非常大，是对全局数据做同样的重做进行数据恢复。

119. Spark RDD 和 MR2 的区别

1）mr2 只有 2 个阶段，数据需要大量访问磁盘，数据来源相对单一 ,spark RDD ,可以 无数个阶段进行迭代计算，数据来源非常丰富，数据落地介质也非常丰富 spark 计算基于内 存；
2）mr2 需要频繁操作磁盘 IO 需要 大家明确的是如果是 SparkRDD 的话，你要知道每一种 数据来源对应的是什么，RDD 从数据源加载数据,将数据放到不同的 partition 针对这些 partition 中的数据进行迭代式计算计算完成之后，落地到不同的介质当中。

120. RDD的数据结构是怎么样的?

一个 RDD 对象，包含如下 5 个核心属性。
1）一个分区列表，每个分区里是 RDD 的部分数据（或称数据块）。
2）一个依赖列表，存储依赖的其他 RDD。
3）一个名为 compute 的计算函数，用于计算 RDD 各分区的值。
4）分区器（可选），用于键/值类型的 RDD，比如某个 RDD 是按散列来分区。
5）计算各分区时优先的位置列表（可选），比如从 HDFS 上的文件生成 RDD 时，RDD 分区的 位置优先选择数据所在的节点，这样可以避免数据移动带来的开销。

121. RDD 算子里操作一个外部 map 比如往里面 put 数据。然后算子外再遍历 map。会有什么 问题吗?

频繁创建额外对象，容易 oom。

122. 怎么用 Spark 做数据清洗?

spark 的 RDD 的转化。

123. Spark 读取数据，是几个 Partition 呢?

从 2 方面介绍和回答，一是说下 partition 是什么，二是说下 partition 如何建的。
1）spark 中的 partion 是弹性分布式数据集 RDD 的最小单元，RDD 是由分布在各个节点上的 partion 组成的。partion 是指的 spark 在计算过程中，生成的数据在计算空间内最小单元， 同一份数据（RDD）的 partion 大小不一，数量不定，是根据 application 里的算子和最初 读入的数据分块数量决定的，这也是为什么叫“弹性分布式”数据集的原因之一。Partition 不会根据文件的偏移量来截取的（比如有 3 个 Partition，1 个是头多少 M 的数据，1 个是 中间多少 M 的数据，1 个是尾部多少 M 的数据），而是从一个原文件这个大的集合里根据某 种计算规则抽取符合的数据来形成一个 Partition 的。
2） 如何创建分区，有两种情况，创建 RDD 时和通过转换操作得到新 RDD 时。对于前者， 在调用 textFile 和 parallelize 方法时候手动指定分区个数即可。例如 sc.parallelize(Array(1, 2, 3, 5, 6), 2) 指定创建得到的 RDD 分区个数为 2。如果没 有指定，partition 数等于 block 数；对于后者，直接调用 repartition 方法即可。实际 上分区的个数是根据转换操作对应多个 RDD 之间的依赖关系来确定，窄依赖子 RDD 由父 RDD 分区个数决定，例如 map 操作，父 RDD 和子 RDD 分区个数一致；Shuffle 依赖则由 分区器（Partitioner）决定，例如 groupByKey(new HashPartitioner(2)) 或者直接 groupByKey(2) 得到的新 RDD 分区个数等于 2。

124. spark-shell提交Spark Application如何解决依赖库

spark-shell的话，利用–driver-class-path选项来指定所依赖的jar文件，注意的是–driver-classpath后如果需要跟着多个jar文件的话，jar文件之间使用冒号(:)来分割。

125. spark-shell 找不到hadoop so问题解决

[main] WARN org.apache.hadoop.util.NativeCodeLoader - Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
在Spark的conf目录下，修改spark-env.sh文件，加入LD_LIBRARY_PATH环境变量，值为 HADOOP的native库路径即可.

126. Spark on Yarn架构是怎么样的？（要会画）

127. Spark on Yarn 模式有哪些优点?

1)与其他计算框架共享集群资源（eg.Spark框架与MapReduce框架同时运行，如果不用Yarn进行资源分配， MapReduce分到的内存资源会很少，效率低下）；资源按需分配，进而提高集群资源利用等。
2)相较于Spark自带的Standalone模式，Yarn的资源分配更加细致。
3)Application部署简化，例如Spark，Storm等多种框架的应用由客户端提交后，由Yarn负责资源的管理和调 度，利用Container作为资源隔离的单位，以它为单位去使用内存,cpu等。
4)Yarn通过队列的方式，管理同时运行在Yarn集群中的多个服务，可根据不同类型的应用程序负载情况，调整 对应的资源使用量，实现资源弹性管理。

128. 介绍一下你对Unified Memory Management内存管理模型的理解?

答：Spark中的内存使用分为两部分：执行（execution）与存储（storage）。执行内存主要用 于shuffles、joins、sorts和aggregations，存储内存则用于缓存或者跨节点的内部数据传输。
1.6之前，对于一个Executor,内存都有哪些部分构成：
1）ExecutionMemory。这片内存区域是为了解决 shuffles,joins, sorts and aggregations 过程 中为了避免频繁IO需要的buffer。 通过spark.shuffle.memoryFraction(默认 0.2) 配置。
2）StorageMemory。这片内存区域是为了解决 block cache(就是你显示调用dd.cache, rdd.persist等方法), 还有就是broadcasts,以及task results的存储。可以通过参数 spark.storage.memoryFraction(默认0.6)。设置。
3）OtherMemory。给系统预留的，因为程序本身运行也是需要内存的。 (默认为0.2).
传统内存管理的不足：
1).Shuffle占用内存0.2*0.8，内存分配这么少，可能会将数据spill到磁盘，频繁的磁盘IO是很大的 负担，Storage内存占用0.6，主要是为了迭代处理。传统的Spark内存分配对操作人的要求非常 高。（Shuffle分配内存：ShuffleMemoryManager, TaskMemoryManager, ExecutorMemoryManager）一个Task获得全部的Execution的 Memory，其他Task过来就没有内存了，只能等待。
2).默认情况下，Task在线程中可能会占满整个内存，分片数据特别大的情况下就会出现这种情 况，其他Task没有内存了，剩下的cores就空闲了，这是巨大的浪费。这也是人为操作的不当造成 的。
3).MEMORY_AND_DISK_SER的storage方式，获得RDD的数据是一条条获取，iterator的方式。 如果内存不够（spark.storage.unrollFraction），unroll的读取数据过程，就是看内存是否足 够，如果足够，就下一条。unroll的space是从Storage的内存空间中获得的。unroll的方式失 败，就会直接放磁盘。
4). 默认情况下，Task在spill到磁盘之前，会将部分数据存放到内存上，如果获取不到内存，就不 会执行。永无止境的等待，消耗CPU和内存。 在此基础上，Spark提出了UnifiedMemoryManager，不再分ExecutionMemory和Storage Memory,实际上还是分的，只不过是Execution Memory访问Storage Memory，Storage Memory也可以访问Execution Memory，如果内存不够，就会去借。

129. 简要描述宽依赖窄依赖以及各自的特点?

rdd 中的--宽依赖---父RDD每个分区的数据可能被多个子RDD分区使用 ，子RDD分区通常对应所有的父RDD分区，这其中分为两种情况：
1,一个父RDD的分区对应所有的子RDD的分区（没有core-patitioned过的join）
2,一个父RDD分区对应非全部的的多个RDD分区（groupByKey）
rdd中的--窄依赖---父RDD每个分区的只被子RDD的一个分区使用，子RDD 通常对应常数个父RDD分区，这其中分为两种情况：1，一个子RDD分区对应一个父RDD分区（filter,map）2,一个子RDD分区对应多个父RDD分区（co-partitioned过的join）
窄依赖与宽依赖的优缺点：
宽依赖----有shuffle----要跨网络拉去数据-----耗资源，窄依赖----一个节点内完成转化-----快速。
当子RDD 需要重算时，宽依赖 会重算所有父RDD分区的数据，这样会出现多余的重算，窄依赖只需计算对应的一个父RDD的数据即可。
窄依赖：filter map flatmap mapPartitions
宽依赖：reduceByKey grupByKey combineByKey，sortByKey, join(no copartition)

130. BlockManager怎么管理硬盘和内存的?

DiskStore：负责磁盘存储
MemoryStore：负责内存存储
BlockManager具体工作原理详见：https://www.jianshu.com/p/d694a3753059

131. Spark on Mesos中，什么是的粗粒度分配，什么是细粒度分配，各自的优点和缺点是什么?

答：1）粗粒度：启动时就分配好资源， 程序启动，后续具体使用就使用分配好的资源，不需要再 分配资源；好处：作业特别多时，资源复用率高，适合粗粒度；不好：容易资源浪费，假如一个 job有1000个task，完成了999个，还有一个没完成，那么使用粗粒度，999个资源就会闲置在那 里，资源浪费。
2）细粒度分配：用资源的时候分配，用完了就立即回收资源，启动会麻烦一点， 启动一次分配一次，会比较麻烦。

132. spark的有几种部署模式，每种模式特点?

1）本地模式 Spark不一定非要跑在hadoop集群，可以在本地，起多个线程的方式来指定。将Spark应用以多 线程的方式直接运行在本地，一般都是为了方便调试，本地模式分三类
• local：只启动一个executor
• local[k]:启动k个executor
• local[*]：启动跟cpu数目相同的 executor。
2)standalone模式分布式部署集群， 自带完整的服务，资源管理和任务监控是Spark自己监控，这个模式也是其他 模式的基础。
3)Spark on yarn模式 分布式部署集群，资源和任务监控交给yarn管理，但是目前仅支持粗粒度资源分配方式，包含 cluster和client运行模式，cluster适合生产，driver运行在集群子节点，具有容错功能，client适 合调试，dirver运行在客户端。
1）Spark On Mesos模式。官方推荐这种模式（当然，原因之一是血缘关系）。正是由于Spark 开发之初就考虑到支持Mesos，因此，目前而言，Spark运行在Mesos上会比运行在YARN上更加 灵活，更加自然。用户可选择两种调度模式之一运行自己的应用程序：
1) 粗粒度模式（Coarse-grained Mode）：每个应用程序的运行环境由一个Dirver和若干个 Executor组成，其中，每个Executor占用若干资源，内部可运行多个Task（对应多少 个“slot”）。应用程序的各个任务正式运行之前，需要将运行环境中的资源全部申请好，且运行 过程中要一直占用这些资源，即使不用，最后程序运行结束后，回收这些资源。
2) 细粒度模式（Fine-grained Mode）：鉴于粗粒度模式会造成大量资源浪费，Spark On Mesos还提供了另外一种调度模式：细粒度模式，这种模式类似于现在的云计算，思想是按需分 配。

133. Spark 中 standalone 模式特点，有哪些优点和缺点?

特点：
1）standalone 是 master/slave 架构，集群由 Master 与 Worker 节点组成，程序通过 与 Master 节点交互申请资源，Worker 节点启动 Executor 运行；
2）standalone 调度模式使 用 FIFO 调度方式；
3）无依赖任何其他资源管理系统，Master 负责管理集群资源
优点：A.部署简单；B.不依赖其他资源管理系统
缺点：A.默认每个应用程序会独占所有可用节点的资源，当然可以通过 spark.cores.max 来决定一个应用可以申请的 CPU cores 个数；
B.可能有单点故障，需要自己配置 master HA

134. 列举你了解的序列化方法，并谈谈序列化有什么好处?

1）序列化：将对象转换为字节流，本质也可以理解为将链表的非连续空间转为连续空间存 储的数组，可以将数据进行流式传输或者块存储，反序列化就是将字节流转为对象。 kyro,Java 的 serialize 等
2）spark 中的序列化常见于
• 进程间通讯：不同节点的数据传输
• 数据持久化到磁盘 在 spark 中扮演非常重要的角色，序列化和反序列化的程度会影响到数据传输速度，甚至影 响集群的传输效率，因此，高效的序列化方法有 2 点好处：a.提升数据传输速度，b.提升数据读写 IO 效率。

135. Spark job的运行架构?

1、构建Spark Application 的运行环境（初始化SparkContext）， SparkContext向资源管理器（可以是Standalone、Mesos或YARN）注册并申请运行Executor资源
2、资源管理器分配 Executor 资源并启动 StandaloneExecutorBackend，Executor 运行情况将随着心跳发送到资源管理器上
3、 SparkContext构建成DAG 图，将DAG 图分解成Stage，并把Taskset发送给TaskScheduler。Executor 向 SparkContext 申请 Task，TaskScheduler 将 Task 发放给 Executor 运行同时 SparkContext将应用程序代码发放Executor
4、Task在 Executor上运行，运行完毕释放所有资源。

136. Spark有几种运行模式

4种：standalone模式，Local模式，Spark on YARN 模式，mesos模式

137. Spark如何指定本地模式

在IDEA，Eclipse里面开发的模式就是Local模式。还有，我们直接用Spark-Shell这种进来的模式也是local。
new SparkConf().setMaster("local")，这时候我们默认的分区数 = 你的本台服务器的CPU core的个数。
分区数 = task 的个数
new SparkConf().setMasater("local[N]") 这个时候分区的个数 = N

138. Spark的算子分类

1）Transformation 变换/转换算子：这种变换并不触发提交作业，完成作业中间过程处理。
Transformation 操作是延迟计算的，也就是说从一个RDD 转换生成另一个 RDD 的转换操作不是马上执行，需要等到有 Action 操作的时候才会真正触发运算。
2）Action 行动算子：这类算子会触发 SparkContext 提交 Job 作业。
Action 算子会触发 Spark 提交作业（Job），并将数据输出 Spark系统。

139. Spark中stage的划分规则

Spark任务会根据RDD之间的依赖关系，形成一个DAG有向无环图，DAG会提交给DAGScheduler，DAGScheduler会把DAG划分相互依赖的多个stage，划分stage的依据就是RDD之间的宽窄依赖。遇到宽依赖就划分stage，每个stage包含一个或多个task任务。然后将这些task以taskSet的形式提交给TaskScheduler运行。stage是由一组并行的task组成。

140. 如何区分spark哪些操作是shuffle操作

Shuffle是划分DAG中stage的标识，同时影响spark执行速度的关键步骤。宽依赖会发生shuffle操作。RDD的Transformtion函数中，又分为窄依赖和宽依赖的操作。窄依赖跟宽依赖的区别是是否发生shuffle操作。窄依赖是子RDD的各个分片不依赖于其他分片，能够独立计算得到结果，宽依赖指子RDD的各个分片会依赖于父RDD的多个分片，会造成父RDD分各个分片在集群中重新分片。

141. RDD，DataFrame，DataSet的区别

DataFrame比RDD多了数据的结构信息，即schema。RDD是分布式的 Java对象的集合。DataFrame是分布式的Row对象的集合。DataFrame除了提供了比RDD更丰富的算子以外，更重要的特点是提升执行效率、减少数据读取以及执行计划的优化；Dataset可以认为是DataFrame的一个特例，主要区别是Dataset每一个record存储的是一个强类型值而不是一个Row

142. 解释一下DAG

DAG，有向无环图，每个操作生成的RDD之间都会连一条线，最后组成的有向无环图

143. map和flatmap的区别

map函数会对每一条输入进行指定的操作，然后为每一条输入返回一个对象；而flatMap函数则是两个操作--先映射后扁平化：同map函数一样：对每一条输入进行指定的操作，然后为每一条输入返回一个对象，最后将所有对象合并为一个对象。

144. spark中rdd.persist()和rdd.cache()的区别

1）RDD的cache()方法其实调用的就是persist方法，缓存策略均为MEMORY_ONLY；
2）可以通过persist方法手工设定StorageLevel来满足工程需要的存储级别；
3）cache或者persist并不是action；

145. spark如何提交脚本

Spark提供了一个容易上手的应用程序部署工具bin/spark-submit，可以完成Spark应用程序在local、Standalone、YARN、Mesos上的快捷部署。可以指定集群资源master，executor/ driver的内存资源等。

146. spark内存不足怎么处理?

1.在不增加内存的情况下，可以通过减少每个Task的大小
2.在shuffle的使用，需要传入一个partitioner，通过设置 spark.default.parallelism参数3.在standalone的模式下如果配置了--total-executor-cores 和 --executor-memory 这两个参数外，还需要同时配置--executor-cores或者spark.executor.cores参数，确保Executor资源分配均匀。
4.如果RDD中有大量的重复数据,或者Array中需要存大量重复数据的时候我们都可以将重复数据转化为String,能够有效的减少内存使用

147. mapreduce和spark计算框架效率区别的原因

spark是借鉴了Mapreduce,并在其基础上发展起来的，继承了其分布式计算的优点并进行了改进，spark生态更为丰富，功能更为强大，性能更加适用范围广，mapreduce更简单，稳定性好。主要区别
（1）spark把运算的中间数据存放在内存，迭代计算效率更高，mapreduce的中间结果需要落地，保存到磁盘
（2）Spark容错性高，它通过弹性分布式数据集RDD来实现高效容错，RDD是一组分布式的存储在节点内存中的只读性的数据集，这些集合石弹性的，某一部分丢失或者出错，可以通过整个数据集的计算流程的血缘关系来实现重建，mapreduce的容错只能重新计算
（3）Spark更通用，提供了transformation和action这两大类的多功能api，另外还有流式处理sparkstreaming模块、图计算等等，mapreduce只提供了map和reduce两种操作，流计算及其他的模块支持比较缺乏。
（4）Spark框架和生态更为复杂，有RDD，血缘lineage、执行时的有向无环图DAG,stage划分等，很多时候spark作业都需要根据不同业务场景的需要进行调优以达到性能要求，mapreduce框架及其生态相对较为简单，对性能的要求也相对较弱，运行较为稳定，适合长期后台运行。

148. Spark数据倾斜怎么处理?

https://tech.meituan.com/spark_tuning_pro.html

149. Spark如何和kafka对接?

1、 KafkaUtils.createStream
执行流程如下:
创建createStream，Receiver被调起执行
连接ZooKeeper，读取相应的Consumer、Topic配置信息等
通过consumerConnector连接到Kafka集群，收取指定topic的数据
创建KafkaMessageHandler线程池来对数据进行处理，通过ReceiverInputDStream中的方法，将数据转换成BlockRDD,供后续计算
2、 KafkaUtils.createDirectStream
执行流程如下:
实例化KafkaCluster，根据用户配置的Kafka参数，连接Kafka集群
通过Kafka API读取Topic中每个Partition最后一次读的Offset
接收成功的数据，直接转换成KafkaRDD,供后续计算

二者区别?:
1.简化的并行：在Receiver的方式中是通过创建多个Receiver之后利用union来合并成一个Dstream的方式提高数据传输并行度。而在Direct方式中，Kafka中的partition与RDD中的partition是一一对应的并行读取Kafka数据，这种映射关系也更利于理解和优化。

2.高效：在Receiver的方式中，为了达到0数据丢失需要将数据存入Write Ahead Log中，这样在Kafka和日志中就保存了两份数据，浪费！而第二种方式不存在这个问题，只要我们Kafka的数据保留时间足够长，我们都能够从Kafka进行数据恢复。

3.精确一次：在Receiver的方式中，使用的是Kafka的高阶API接口从Zookeeper中获取offset值，这也是传统的从Kafka中读取数据的方式，但由于Spark Streaming消费的数据和Zookeeper中记录的offset不同步，这种方式偶尔会造成数据重复消费。而第二种方式，直接使用了简单的低阶Kafka API，Offsets则利用Spark Streaming的checkpoints进行记录，消除了这种不一致性。

150. Spark如何和mysql对接?

实例化Properties，调用put方法，根据用户配置的mysql的参数填写put方法的参数，编写jdbc的链接的URL的信息，使用实例化的SQLContext调用read的jbdc方法，生成一个DataFrame ,供后续计算

151. 用过spark和kafka组合吗？Spark阻塞了，是什么原因导致的?

spark streaming虽然是按照时间片消费数据的，但是上一个批次的数据没有处理完，下一个批次也会继续处理，如果有很多批次的数据同时在处理。会拖垮集群，服务器的资源会使用频繁，导致很慢，甚至任务阻塞。
看看时间片是否过小，最小的时间间隔，参考在0.5~2秒钟之间。可以适当放宽时间片的大小。

152. Scala隐式转换

Scala中的隐式转换是一种非常强大的代码查找机制。当函数、构造器调用缺少参数时或者某一实例调用了其他类型的方法导致编译不通过时，编译器会尝试搜索一些特定的区域，尝试使编译通过。

```
触发时机：
1）当一个对象去调用某个方法的时候，不具备这个方法
2）当某个对象去调用某个方法，但是传入的参数类型不匹配
3）A <% B       // <% 是把A类型转化成B类型的意思
```

A必须是B的子类,但是如果A如果不是B的子类，那么这个时候会触发隐式转换，把A变为B的类型

①在Scala当中,要想使用隐式转换,必须标记为implicit关键字,implicit关键字可以用来修饰参数(隐式值与隐式参数)、函数(隐式视图)、类(隐式类)、对象(隐式对象).
②隐式转换在整个作用域中,必须是单一的标识符,进而避免隐式冲突.
③在隐式转换的作用域查找中，如果当前作用域没有隐式转换，编译器就会自动到相应源或目标类型的伴生对象中查找隐式转换。
④Scala中的隐式值、隐式类、隐式对象常放在单例对象object中.

153. Scala高级函数

可以使用其他函数作为参数，或者使用函数作为输出结果，会自动类型推断，比如 map、foreach

154. Spark整合Hive

1.添加对应jar包依赖
2.集群上启动 hivesever2 服务
nohup hiveserver2 1>/dev/null 2>/dev/null &
或者 metastore 服务
metastore需要添加如下配置：

hive.metastore.uris
thrift://hadoop4:9083

开启服务：nohup hive --service metastore 1>/dev/null 2>/dev/null &
3.将hive-site.xml放进classpath
4.sparkSession=SparkSession.builder().master("local").appName("testhive").enableHiveSupport().getOrCreate() //一定要调用一下enableHiveSupport()

155. (重要）手写 spark 求 π、wordcount、二次排序

一定要会！！！

156. scala 中 private 与 private[this] 修饰符的区别?

1）private ，类私有的字段，Scala 会自动生成私有的 getter/setter 方法，通过对象实例可以调用如下面的 other.job；
2）private[this]，对象私有的字段，Scala 不生成 getter/setter 方法，所以只能在对象内部访问被修饰的字段，如下代码是不能编译通过的，因为没有生成 getter/setter 方法，所以不能通过这个方法调用。private[this]比 private 要更加严格，他将声明的变量只能在自己的同一个实例中可以被访问。

157. scala 中内部类和 java 中的内部类区别

1)scala 内部类：同样的类的内部类的不同实例，属于不同的类型
内部类纯属于对象的(属于外部类的实例本身)，比如构造内部类对象方法
scala:val test=new 外部类名.内部类构造方法
2）java 内部类：java 内部类是一个编译时的概念，一旦编译成功，就会成为完全不同的两类。对于一个名为 outer 的外部类和其内部定义的名为 inner 的内部类。编译完成后出现 outer.class 和 outer$inner.class 两类。所以内部类的成员变量/方法名可以和外部类的相同。

158. Spark 中 standalone 模式特点，有哪些优点和缺点?

特点：1)standalone 是 master/slave 架构，集群由 Master 与 Worker 节点组成，程序通过与 Master 节点交互申请资源，Worker 节点启动 Executor 运行；2）standalone 调度模式使用 FIFO 调度方式；3）无依赖任何其他资源管理系统，Master 负责管理集群资源
优点：1）部署简单；2）不依赖其他资源管理系统
缺点： 1）默认每个应用程序会独占所有可用节点的资源，当然可以通过 spark.cores.max 来决定一个应用可以申请的 CPU cores 个数；2）可能有单点故障，需要自己配置 master HA

159. FIFO 调度模式的基本原理、优点和缺点?

基本原理：按照先后顺序决定资源的使用，资源优先满足最先来的 job。第一个 job 优先获取所有可用的资源，接下来第二个 job 再获取剩余资源。以此类推，如果第一个 job 没有占用所有的资源，那么第二个 job 还可以继续获取剩余资源，这样多个 job 可以并行运行，如果第一个 job 很大，占用所有资源，则第二 job 就需要等待，等到第一个 job 释放所有资源。
优点和缺点：1)适合长作业，不适合短作业,2)适合 CPU 繁忙型作业（计算时间长，相当于长作业），不利于 IO 繁忙型作业（计算时间短，相当于短作业）

160. FAIR 调度模式的优点和缺点?

所有的任务拥有大致相当的优先级来共享集群资源，spark 多以轮训的方式为任务分配资源，不管长任务还是端任务都可以获得资源，并且获得不错的响应时间，对于短任务，不会像 FIFO 那样等待较长时间了，通过参数 spark.scheduler.mode 为 FAIR 指定.

161. CAPCACITY 调度模式的优点和缺点?

原理：
计算能力调度器支持多个队列，每个队列可配置一定的资源量，每个队列采用 FIFO 调度策略，为了防止同一个用户的作业独占队列中的资源，该调度器会对同一用户提交的作业所占资源量进行限定。调度时，首先按以下策略选择一个合适队列：计算每个队列中正在运行的任务数与其应该分得的计算资源之间的比值(即比较空闲的队列)，选择一个该比值最小的队列；然后按以下策略选择该队列中一个作业：按照作业优先级和提交时间顺序选择，同时考虑用户资源量限制和内存限制
优点：
1) 计算能力保证。支持多个队列，某个作业可被提交到某一个队列中。每个队列会配置一定比例的计算资源，且所有提交到队列中的作业共享该队列中的资源。
(2) 灵活性。空闲资源会被分配给那些未达到资源使用上限的队列，当某个未达到资源的队列需要资源时，一旦出现空闲资源资源，便会分配给他们。
(3) 支持优先级。队列支持作业优先级调度（默认是 FIFO）
(4) 多重租赁。综合考虑多种约束防止单个作业、用户或者队列独占队列或者集群中的资源。
(5) 基于资源的调度。 支持资源密集型作业，允许作业使用的资源量高于默认值，进而可容纳不同资源需求的作业。不过，当前仅支持内存资源的调度。

162. 列举你了解的序列化方法，并谈谈序列化有什么好处?

1）序列化：将对象转换为字节流，本质也可以理解为将链表的非连续空间转为连续空间存储的数组，可以将数据进行流式传输或者块存储，反序列化就是将字节流转为对象。
kyro,Java 的 serialize 等
2）spark 中的序列化常见于
· 进程间通讯：不同节点的数据传输
· 数据持久化到磁盘
在 spark 中扮演非常重要的角色，序列化和反序列化的程度会影响到数据传输速度，甚至影响集群的传输效率，因此，高效的序列化方法有 2 点好处：a.提升数据传输速度，b.提升数据读写 IO 效率。

163. 常见的数压缩方式，你们生产集群采用了什么压缩方式，提升了多少效率?

1)数据压缩，大片连续区域进行数据存储并且存储区域中数据重复性高的状况下，可以使用适当的压缩算法。数组，对象序列化后都可以使用压缩，数更紧凑，减少空间开销。常见的压缩方式有 snappy，LZO，gz 等
2)Hadoop 生产环境常用的是 snappy 压缩方式（使用压缩，实际上是 CPU 换 IO 吞吐量和磁盘空间，所以如果 CPU 利用率不高，不忙的情况下，可以大大提升集群处理效率）。snappy 压缩比一般 20%~30%之间，并且压缩和解压缩效率也非常高（参考数据如下）。
a.GZIP 的压缩率最高，但是其实 CPU 密集型的，对 CPU 的消耗比其他算法要多，压缩和解压速度也慢；b.LZO 的压缩率居中，比 GZIP 要低一些，但是压缩和解压速度明显要比 GZIP 快很多，其中解压速度快的更多；
c.Zippy/Snappy 的压缩率最低，而压缩和解压速度要稍微比 LZO 要快一些。
提升了多少效率可以从 2 方面回答，1）数据存储节约多少存储，2）任务执行消耗时间节约了多少，可以举个实际例子展开描述。

164. 简要描述 Spark 写数据的流程?

1）RDD 调用 compute 方法，进行指定分区的写入
2）CacheManager 中调用 BlockManager 判断数据是否已经写入，如果未写，则写入
3）BlockManager 中数据与其他节点同步
4）BlockManager 根据存储级别写入指定的存储层
5）BlockManager 向主节点汇报存储状态中

165. Spark 中 Lineage 的基本原理

这里应该是问你 Spark 的容错机制的原理：
1）Lineage（又称为 RDD 运算图或 RDD 依赖关系图）是 RDD 所有父 RDD 的 graph（图）。它是在 RDD 上执行 transformations 函数并创建 logical execution plan（逻辑执行计划）的结果，是 RDD 的逻辑执行计划，记录了 RDD 之间的依赖关系。
2） 使用 Lineage 实现 spark 的容错，本质上类似于数据库中重做日志，是容错机制的一种方式，不过这个重做日志粒度非常大，是对全局数据做同样的重做进行数据恢复。

166. 使用 shell 和 scala 代码实现 WordCount?

这个题目即考察了你对 shell 的掌握，又考察了你对 scala 的了解，还考察了你动手写代码的能力，是比较好的一道题（实际开发中，有些代码是必须要背下来的，烂熟于心，劣等的程序员就是百度+copy，是不可取的）

167. scala 版本（spark）的 wordCount

val conf = new SparkConf() val sc = new SparkContext(conf) val line = sc.textFile("xxxx.txt") line.flatMap(*.split(" ")).map((*,1)).reduceByKey(+).collect().foreach(println)

sc.stop()

168. 请列举你碰到的 CPU 密集型的应用场景，你有做哪些优化?

1）CPU 密集型指的是系统的 硬盘/内存 效能 相对 CPU 的效能 要好很多，此时，系统运作，大部分的状况是 CPU Loading 100%，CPU 要读/写 I/O (硬盘/内存)，I/O 在很短的时间就可以完成，而 CPU 还有许多运算要处理，CPU Loading 很高。->cpu 是瓶颈
I/O 密集型指的是系统的 CPU 效能相对硬盘/内存的效能要好很多，此时，系统运作，大部分的状况是 CPU 在等 I/O (硬盘/内存) 的读/写，此时 CPU Loading 不高。->IO 是瓶颈
2）CPU 密集型主要特点是要进行大量的计算，常见应用场景有：图计算、大量的逻辑判断程序，机器学习等，Mahout 其实就是针对 CPU 密集的一个 apache 项目。
优化的点主要有，1）降低任务的并行执行，务越多，花在任务切换的时间就越多，CPU 执行任务的效率就越低，2）优化计算逻辑，减少计算逻辑的复杂度，3）尽量减少使用高强度压缩方式，对原始数据的压缩和解压缩会增加 CPU 的负担

169. Spark RDD 和 MR2 的区别

1）mr2 只有 2 个阶段，数据需要大量访问磁盘，数据来源相对单一 ,spark RDD ,可以无数个阶段进行迭代计算，数据来源非常丰富，数据落地介质也非常丰富 spark 计算基于内存，
2）mr2 需要频繁操作磁盘 IO 需要 大家明确的是如果是 SparkRDD 的话，你要知道每一种数据来源对应的是什么，RDD 从数据源加载数据,将数据放到不同的 partition 针对这些 partition 中的数据进行迭代式计算计算完成之后，落地到不同的介质当中
14.Spark 读取 hdfs 上的文件，然后 count 有多少行的操作，你可以说说过程吗。那这个 count 是在内存中，还是磁盘中计算的呢?
1）从任务执行的角度分析执行过程
driver生成逻辑执行计划->driver生成物理执行计划->driver任务调度->executor任务执行 。
·四个阶段
逻辑执行计划-》成物理执行计划-》任务调度-》任务执行
·四个对象
driver-》DAGScheduler-》TaskScheduler-》Executor
·两种模式
任务解析、优化和提交单机模式-》任务执行分布式模式
2）计算过程发生在内存

170. Spark 和 Mapreduce 快？ 为什么快呢？ 快在哪里呢?

Spark 更加快的主要原因有几点：1）基于内存计算，减少低效的磁盘交互；2）高效的调度算法，基于 DAG;3)容错机制 Lingage，主要是 DAG 和 Lianage，及时 spark 不使用内存技术，也大大快于 mapreduce

171. Spark sql 又为什么比 hive 快呢?

计算引擎不一样，一个是 spark 计算模型，一个是 mapreudce 计算模型

172. RDD 的数据结构是怎么样的?

个 RDD 对象，包含如下 5 个核心属性。
1）一个分区列表，每个分区里是 RDD 的部分数据（或称数据块）。
2）一个依赖列表，存储依赖的其他 RDD。
3）一个名为 compute 的计算函数，用于计算 RDD 各分区的值。
4）分区器（可选），用于键/值类型的 RDD，比如某个 RDD 是按散列来分区。
5）计算各分区时优先的位置列表（可选），比如从 HDFS 上的文件生成 RDD 时，RDD 分区的位置优先选择数据所在的节点，这样可以避免数据移动带来的开销。

173. RDD 算子里操作一个外部 map 比如往里面 put 数据。然后算子外再遍历 map。会有什么

问题吗。
频繁创建额外对象，容易 oom

174. hadoop 的生态呢。说说你的认识。

hadoop 生态主要分为三大类型，1）分布式文件系统，2）分布式计算引擎，3）周边工具
1）分布式系统：HDFS，hbase
2）分布式计算引擎：Spark，MapReduce
3）周边工具：如 zookeeper，pig，hive，oozie，sqoop，ranger，kafka 等
可以参考介绍 http://www.cnblogs.com/zhijianliutang/articles/5195045.html

175. jvm 怎么调优的，介绍你的 Spark JVM 调优经验?

参考梅峰谷博文 《Spark 应用经验与程序调优》设置合理的 JVM 部分

176. jvm 结构？堆里面几个区?

1）JVM 内存区域分为方法去、虚拟机栈、本地方法栈、堆、程序计数器
方法区：也称"永久代”、“非堆”， 它用于存储虚拟机加载的类信息、常量、静态变量、是各个线程共享的内存区域
虚拟机栈：描述的是 Java 方法执行的内存模型：每个方法被执行的时候 都会创建一个“栈帧”用于存储局部变量表(包括参数)、操作栈、方法出口等信息
本地方法栈：与虚拟机栈基本类似，区别在于虚拟机栈为虚拟机执行的 java 方法服务，而本地方法栈则是为 Native 方法服务
堆：也叫做 java 堆、GC 堆是 java 虚拟机所管理的内存中最大的一块内存区域，也是被各个线程共享的内存区域，在 JVM 启动时创建。
程序计数器：是最小的一块内存区域，它的作用是当前线程所执行的字节码的行号指示器。
2）堆：a.JVM 中共享数据空间可以分成三个大区，新生代（Young Generation）、老年代（Old Generation）、永久代（Permanent Generation），其中 JVM 堆分为新生代和老年代，b.新生代可以划分为三个区，Eden 区（存放新生对象），两个幸存区（From Survivor 和 To Survivor）（存放每次垃圾回收后存活的对象）；c.永久代管理 class 文件、静态对象、属性等（JVM uses a separate region of memory, called the Permanent Generation (orPermGen for short), to hold internal representations of java classes. PermGen is also used to store more information ）；d.JVM 垃圾回收机制采用“分代收集”：新生代采用复制算法，老年代采用标记清理算法
备注:对虚拟机不了解的朋友，可以看梅峰谷分享的 JVM 虚拟机系列视频教程和书籍
链接：https://pan.baidu.com/s/1c1Rw79e 密码：6qof

177. 怎么用 Spark 做数据清洗

spark 的 RDD 的转化

178. Spark 怎么整合 hive?

1).将 hive 的配置文件 hive-site.xml 复制到 Spark conf 目录下
2）根据 hive 的配置参数 hive.metastore.uris 的情况，采用不同的集成方式
a. jdbc 方式：hive.metastore.uris 没有给定配置值，为空(默认情况),SparkSQL 通过 hive 配置的 javax.jdo.option.XXX 相关配置值直接连接metastore 数据库直接获取hive 表元数据, 需要将连接数据库的驱动添加到 Spark 应用的 classpath 中
b.metastore服务方式：hive.metastore.uris给定了具体的参数值,SparkSQL通过连接hive 提供的 metastore 服务来获取 hive 表的元数据, 直接启动 hive 的 metastore 服务即可完成 SparkSQL 和 Hive 的集成:
3)使用 metastore 服务方式，对 hive-site.xml 进行配置

hive.metastore.uris
 trhift://mfg-hadoop:9083

4).启动 hive service metastore 服务
bin/hive --service metastore &
5)启动 spark-sql 测试，执行 show databases 命令，检查是不是和 hive 的数据库一样的。

179. Spark 读取数据，是几个 Partition 呢?

答：从 2 方面介绍和回答，一是说下 partition 是什么玩意，二是说下 partition 如何建的
1）spark 中的 partion 是弹性分布式数据集 RDD 的最小单元，RDD 是由分布在各个节点上的 partion 组成的。partion 是指的 spark 在计算过程中，生成的数据在计算空间内最小单元，同一份数据（RDD）的 partion 大小不一，数量不定，是根据 application 里的算子和最初读入的数据分块数量决定的，这也是为什么叫"弹性分布式"数据集的原因之一。Partition 不会根据文件的偏移量来截取的（比如有 3 个 Partition，1 个是头多少 M 的数据，1 个是中间多少 M 的数据，1 个是尾部多少 M 的数据），而是从一个原文件这个大的集合里根据某种计算规则抽取符合的数据来形成一个 Partition 的
2） 如何创建分区，有两种情况，创建 RDD 时和通过转换操作得到新 RDD 时。对于前者，在调用 textFile 和 parallelize 方法时候手动指定分区个数即可。例如 sc.parallelize(Array(1, 2, 3, 5, 6), 2) 指定创建得到的 RDD 分区个数为 2。如果没有指定，partition 数等于 block 数；对于后者，直接调用 repartition 方法即可。实际上分区的个数是根据转换操作对应多个 RDD 之间的依赖关系来确定，窄依赖子 RDD 由父 RDD 分区个数决定，例如 map 操作，父 RDD 和子 RDD 分区个数一致；Shuffle 依赖则由分区器（Partitioner）决定，例如 groupByKey(new HashPartitioner(2)) 或者直接 groupByKey(2) 得到的新 RDD 分区个数等于 2。

180. hbase region 多大会分区，Spark 读取 hbase 数据是如何划分 partition 的?

答：region 超过了 hbase.hregion.max.filesize 这个参数配置的大小就会自动裂分，默认值是 1G。
默认情况下，hbase 有多少个 region，Spark 读取时就会有多少个 partition

181. 画图，画 Spark 的工作模式，部署分布架构图

参考梅峰谷博文《【Spark 你妈喊你回家吃饭-11】Spark 基本概念和运行模式》

182. 画图，画图讲解 Spark 工作流程。以及在集群上和各个角色的对应关系。

参考梅峰谷博文《【Spark 你妈喊你回家吃饭-13】Spark 计算引擎剖析》

183. Java 自带有哪几种线程池。

1)newCachedThreadPool
创建一个可缓存线程池，如果线程池长度超过处理需要，可灵活回收空闲线程，若无可回收，则新建线程。这种类型的线程池特点是：
 工作线程的创建数量几乎没有限制(其实也有限制的,数目为 Interger. MAX_VALUE), 这样可灵活的往线程池中添加线程。
 如果长时间没有往线程池中提交任务，即如果工作线程空闲了指定的时间(默认为 1 分钟)，则该工作线程将自动终止。终止后，如果你又提交了新的任务，则线程池重新创建一个工作线程。
 在使用 CachedThreadPool 时，一定要注意控制任务的数量，否则，由于大量线程同时运行，很有会造成系统瘫痪。
2)newFixedThreadPool

创建一个指定工作线程数量的线程池。每当提交一个任务就创建一个工作线程，如果工作线程数量达到线程池初始的最大数，则将提交的任务存入到池队列中。FixedThreadPool 是一个典型且优秀的线程池，它具有线程池提高程序效率和节省创建线程时所耗的开销的优点。但是，在线程池空闲时，即线程池中没有可运行任务时，它不会释放工作线程，还会占用一定的系统资源。
3)newSingleThreadExecutor
创建一个单线程化的 Executor，即只创建唯一的工作者线程来执行任务，它只会用唯一的工作线程来执行任务，保证所有任务按照指定顺序(FIFO, LIFO, 优先级)执行。如果这个线程异常结束，会有另一个取代它，保证顺序执行。单工作线程最大的特点是可保证顺序地执行各个任务，并且在任意给定的时间不会有多个线程是活动的。
4)newScheduleThreadPool
创建一个定长的线程池，而且支持定时的以及周期性的任务执行，支持定时及周期性任务执行。延迟 3 秒执行.

184. 画图，讲讲 shuffle 的过程。那你怎么在编程的时候注意避免这些性能问题

参考梅峰谷博文《【Spark 你妈喊你回家吃饭-13】Spark 计算引擎剖析》 Spark Shuffle解析部分

185. BlockManager 怎么管理硬盘和内存的

参考王家林的视频教程《38-BlockManager 架构原理、运行流程图和源码解密》

186. 关于你简介中的通读Spark源码，它源码大概有多少行，通读又是阅读了什么内容?

回答：Spark最新的1.6.x版本源码有50万行，通读了Spark资源调度层面，Spark计算流程相关的代码（例如
DAGScheduler，TaskSchedulerImpl,RDD等与Spark计算相关性较大的源码块，但是关于WebUI，listenerBus监听器等部分
内容由于在计算框架中使用不多，还没有去阅读）

187. 对于Spark进行数据挖掘计算，你有什么看法?

回答：Spark在1.3.0以后出现的DataFrame可以对结构化的数据进行类SQL语句的数据挖掘(问：那么Spark如何处理非结
构化数据？(回答：通过Scala的函数式编程进行基于RDD的非结构化数据处理))

188. Spark性能优化主要有哪些手段?

回答：
将默认调用的java序列化器改为kyro序列化器(减少序列化数据80%的空间占用(问：为何序列化可以减少存储空间占用
(回答：对同类型的数据对象的头进行压缩合并：我感觉这个我回答错误了)))；
由于Spark1.6.0的统一内存管理模型，若算法的数据量大，而计算逻辑较为简单，可以增大内存管理中cache块的比例
(默认70%(我也一下子想不起来，感觉这个附近))，如果是数据量小而算法逻辑复杂，可以适当减少cache快的比例；
如果因对是集群CPU资源过分盈余，可以采用增加core的数目，但是core的数目增加到一定程度后，依旧无法完全利用
CPU的计算资源，可以选择增加Executor的数目，通常环境下，一个Executor推荐设置5个Core的个数，超过5个推荐增
加Executor的个数
面试官没有询问关于OOM情况的优化(后面了解，他们目前还是处于测试环境，集群性能相对优越，这方面应该还没有
碰到情况)

189. 简要描述Spark分布式集群搭建的步骤

回答：创建相关用户，获得文件权限，建立SSH免密码通信，下载各软件包，解压缩后配置环境变量，之后配置
Hadoop配置文件(hdfs-site.xml，yarn资源管理器相关内容)，Spark配置文件(集群的MasterIP,各节点的内存值)
第三类问题：算法方向使用经验和设计经验
1.简要描述你了解的一些数据挖掘算法与内容
回答：讲解了关于朴素贝叶斯和关联规则这两个基本算法的原理，并举了一个案例(提问关于案例的使用时网上的案例
还是你实际情况使用的案例，我说的是老师给我的一个案例，就不是网上的案例)
2.描述你使用数据挖掘算法的一些实例场景
回答：关于基于文本数据的文本情感分析。。。。
3.是否尝试过基于现实中的某种问题设计一个算法
回答：尝试过关于一个球队建模，不过算法非常简单，然后被面试官说这个内容需要长时间积累。

190. Spark 的四大组件下面哪个不是 ( )

A.Spark Streaming B Mlib
C Graphx D Spark R

191. 下面哪个端口不是 spark 自带服务的端口 ( )

A.8080 B.4040 C.8090 D.18080

192. spark 1.4 版本的最大变化 ( )

A spark sql Release 版本 B 引入 Spark R
C DataFrame D支持动态资源分配

193. Spark Job 默认的调度模式 ( )

A FIFO B FAIR C 无 D 运行时指定

194. 哪个不是本地模式运行的个条件 ( )

A spark.localExecution.enabled=true B 显式指定本地运行 C finalStage 无父 Stage D partition默认值

195. 下面哪个不是 RDD 的特点 ( )

A. 可分区 B 可序列化 C 可修改 D 可持久化

196. 关于广播变量，下面哪个是错误的 ( )

A 任何函数调用 B 是只读的 C 存储在各个节点 D 存储在磁盘或 HDFS

197. 关于累加器，下面哪个是错误的 ( )

A 支持加法 B 支持数值类型
C 可并行 D 不支持自定义类型

198. Spark 支持的分布式部署方式中哪个是错误的 ( )

A standalone B spark on mesos
C spark on YARN D Spark on local

199. Stage 的 Task 的数量由什么决定 ( )

A Partition B Job C Stage D TaskScheduler

200. 下面哪个操作是窄依赖 ( )

A join B filter
C group D sort

201. 下面哪个操作肯定是宽依赖 ( )

A map B flatMap
C reduceByKeyD sample

202. spark 的 master 和 worker 通过什么方式进行通信的? ( )

A http B nio C nettyD Akka

203. 默认的存储级别 ( )

A MEMORY_ONLY B MEMORY_ONLY_SER
C MEMORY_AND_DISK D MEMORY_AND_DISK_SER

204. spark.deploy.recoveryMode 不支持那种 ( )

A.ZooKeeper B. FileSystem
D NONE D Hadoop

205. 下列哪个不是 RDD 的缓存方法 ( )

A persist() B Cache()
C Memory()

206. Task 运行在下来哪里个选项中 Executor 上的工作单元 ( )

A Driver program B. spark master
C.worker node D Cluster manager

207. Hive 的元数据存储在 derby 和 MySQL 中有什么区别 ( )

A.没区别 B.多会话 C.支持网络环境 D数据库的区别

208. DataFrame 和 RDD 最大的区别 ( )

A.科学统计支持 B.多了 schema
C.存储方式不一样 D.外部数据源支持

209. Master 的 ElectedLeader 事件后做了哪些操作 ( )

A. 通知 driver B.通知 worker
C.注册 application D.直接 ALIVE
答案：
DCBAD CDDDA
BCDAD CCBBD

210. 简答说一下Hadoop的map-reduce编程模型

211. hadoop的TextInputFormat作用是什么，如何自定义实现

212. hadoop和Spark的都是并行计算，那么他们有什么相同和区别

213. 为什么要用flume导入hdfs，hdfs的构架是怎样的

214. map-reduce程序运行的时候会有什么比较常见的问题

215. 简单说一下hadoop和spark的shuffle过程

以下是自己的理解，如果有不对的地方希望各位大侠可以帮我指出来~:

216. 简答说一下hadoop的map-reduce编程模型

首先map task会从本地文件系统读取数据，转换成key-value形式的键值对集合
使用的是hadoop内置的数据类型，比如longwritable、text等
将键值对集合输入mapper进行业务处理过程，将其转换成需要的key-value在输出
之后会进行一个partition分区操作，默认使用的是hashpartitioner，可以通过重写hashpartitioner的getpartition方法来自定义分区规则
之后会对key进行进行sort排序，grouping分组操作将相同key的value合并分组输出，在这里可以使用自定义的数据类型，重写WritableComparator的
Comparator方法来自定义排序规则，重写RawComparator的compara方法来自定义分组规则
之后进行一个combiner归约操作，其实就是一个本地段的reduce预处理，以减小后面shufle和reducer的工作量
reducetask会通过网络将各个数据收集进行reduce处理，最后将数据保存或者显示，结束整个job

217. hadoop的TextInputFormat作用是什么，如何自定义实现

InputFormat会在map操作之前对数据进行两方面的预处理
1是getSplits，返回的是InputSplit数组，对数据进行split分片，每片交给map操作一次
2是getRecordReader，返回的是RecordReader对象，对每个split分片进行转换为key-value键值对格式传递给map
常用的InputFormat是TextInputFormat，使用的是LineRecordReader对每个分片进行键值对的转换，以行偏移量作为键，行内容作为值
自定义类继承InputFormat接口，重写createRecordReader和isSplitable方法
在createRecordReader中可以自定义分隔符

218. hadoop和spark的都是并行计算，那么他们有什么相同和区别

两者都是用mr模型来进行并行计算，hadoop的一个作业称为job，job里面分为map task和reducetask，每个task都是在自己的进程中运行的，当task结束
时，进程也会结束
spark用户提交的任务成为application，一个application对应一个sparkcontext，app中存在多个job，每触发一次action操作就会产生一个job
这些job可以并行或串行执行，每个job中有多个stage，stage是shuffle过程中DAGSchaduler通过RDD之间的依赖关系划分job而来的，每个stage里面有多
个task，组成taskset有TaskSchaduler分发到各个executor中执行，executor的生命周期是和app一样的，即使没有job运行也是存在的，所以task可以快速启
动读取内存进行计算
hadoop的job只有map和reduce操作，表达能力比较欠缺而且在mr过程中会重复的读写hdfs，造成大量的io操作，多个job需要自己管理关系
spark的迭代计算都是在内存中进行的，API中提供了大量的RDD操作如join，groupby等，而且通过DAG图可以实现良好的容错

219. 为什么要用flume导入hdfs，hdfs的构架是怎样的

flume可以实时的导入数据到hdfs中，当hdfs上的文件达到一个指定大小的时候会形成一个文件，或者超过指定时间的话也形成一个文件
文件都是存储在datanode上面的，namenode记录着datanode的元数据信息，而namenode的元数据信息是存在内存中的，所以当文件切片很小或者很多
的时候会卡死

220. map-reduce程序运行的时候会有什么比较常见的问题

比如说作业中大部分都完成了，但是总有几个reduce一直在运行
这是因为这几个reduce中的处理的数据要远远大于其他的reduce，可能是因为对键值对任务划分的不均匀造成的数据倾斜
解决的方法可以在分区的时候重新定义分区规则对于value数据很多的key可以进行拆分、均匀打散等处理，或者是在map端的combiner中进行数据预处
理的操作

221. 简单说一下hadoop和spark的shuffle过程

hadoop：map端保存分片数据，通过网络收集到reduce端
spark：spark的shuffle是在DAGSchedular划分Stage的时候产生的，TaskSchedule要分发Stage到各个worker的executor
减少shuffle可以提高性能
部分答案不是十分准确欢迎补充:-)
——-补充更新———

222. Hive中存放是什么?

表。
存的是和hdfs的映射关系，hive是逻辑上的数据仓库，实际操作的都是hdfs上的文件，HQL就是用sql语法来写的mr程序。

223. Hive与关系型数据库的关系?

没有关系，hive是数据仓库，不能和数据库一样进行实时的CURD操作。
是一次写入多次读取的操作，可以看成是ETL工具。

224. Flume工作机制是什么?

核心概念是agent，里面包括source、chanel和sink三个组件。
source运行在日志收集节点进行日志采集，之后临时存储在chanel中，sink负责将chanel中的数据发送到目的地。
只有成功发送之后chanel中的数据才会被删除。
首先书写flume配置文件，定义agent、source、chanel和sink然后将其组装，执行flume-ng命令。

225. Sqoop工作原理是什么?

hadoop生态圈上的数据传输工具。
可以将关系型数据库的数据导入非结构化的hdfs、hive或者bbase中，也可以将hdfs中的数据导出到关系型数据库或者文本文件中。
使用的是mr程序来执行任务，使用jdbc和关系型数据库进行交互。
import原理：通过指定的分隔符进行数据切分，将分片传入各个map中，在map任务中在每行数据进行写入处理 没有reduce。
export原理：根据要操作的表名生成一个Java类，并读取其元数据信息和分隔符对非结构化的数据进行匹配，多个map作业同时执行写入关系型数据
库

226. Hbase行健列族的概念，物理模型，表的设计原则?

行健：是hbase表自带的，每个行健对应一条数据。
列族：是创建表时指定的，为列的集合，每个列族作为一个文件单独存储，存储的数据都是字节数组，其中的数据可以有很多，通过时间戳来区
分。
物理模型：整个hbase表会拆分为多个region，每个region记录着行健的起始点保存在不同的节点上，查询时就是对各个节点的并行查询，当region很大
时使用.META表存储各个region的起始点，-ROOT又可以存储.META的起始点。
rowkey的设计原则：各个列簇数据平衡，长度原则、相邻原则，创建表的时候设置表放入regionserver缓存中，避免自动增长和时间，使用字节数组代
替string，最大长度64kb，最好16字节以内，按天分表，两个字节散列，四个字节存储时分毫秒。
列族的设计原则：尽可能少（按照列族进行存储，按照region进行读取，不必要的io操作），经常和不经常使用的两类数据放入不同列族中，列族名
字尽可能短。

227. Spark Streaming和Storm有何区别?

一个实时毫秒一个准实时亚秒，不过storm的吞吐率比较低。

228. mllib支持的算法?

大体分为四大类，分类、聚类、回归、协同过滤。

229. 简答说一下hadoop的map-reduce编程模型?

首先map task会从本地文件系统读取数据，转换成key-value形式的键值对集合。
将键值对集合输入mapper进行业务处理过程，将其转换成需要的key-value在输出。
之后会进行一个partition分区操作，默认使用的是hashpartitioner，可以通过重写hashpartitioner的getpartition方法来自定义分区规则。
之后会对key进行进行sort排序，grouping分组操作将相同key的value合并分组输出。
在这里可以使用自定义的数据类型，重写WritableComparator的Comparator方法来自定义排序规则，重写RawComparator的compara方法来自定义分组规
则。
之后进行一个combiner归约操作，其实就是一个本地段的reduce预处理，以减小后面shufle和reducer的工作量。
reducetask会通过网络将各个数据收集进行reduce处理，最后将数据保存或者显示，结束整个job。

230. Hadoop平台集群配置、环境变量设置?

zookeeper：修改zoo.cfg文件，配置dataDir，和各个zk节点的server地址端口，tickTime心跳时间默认是2000ms，其他超时的时间都是以这个为基础的整
数倍，之后再dataDir对应目录下写入myid文件和zoo.cfg中的server相对应。
hadoop：修改
hadoop-env.sh配置java环境变量
core-site.xml配置zk地址，临时目录等
hdfs-site.xml配置nn信息，rpc和http通信地址，nn自动切换、zk连接超时时间等
yarn-site.xml配置resourcemanager地址
mapred-site.xml配置使用yarn
slaves配置节点信息

格式化nn和zk。
hbase：修改
hbase-env.sh配置java环境变量和是否使用自带的zk
hbase-site.xml配置hdfs上数据存放路径，zk地址和通讯超时时间、master节点
regionservers配置各个region节点
zoo.cfg拷贝到conf目录下
spark：
安装Scala
修改spark-env.sh配置环境变量和master和worker节点配置信息
环境变量的设置：直接在/etc/profile中配置安装的路径即可，或者在当前用户的宿主目录下，配置在.bashrc文件中，该文件不用source重新打开shell窗口即可，配置在.bash_profile的话只对当前用户有效。

231. Hadoop性能调优?

调优可以通过系统配置、程序编写和作业调度算法来进行。
hdfs的block.size可以调到128/256（网络很好的情况下，默认为64）
调优的大头：mapred.map.tasks、mapred.reduce.tasks设置mr任务数（默认都是1）
mapred.tasktracker.map.tasks.maximum每台机器上的最大map任务数
mapred.tasktracker.reduce.tasks.maximum每台机器上的最大reduce任务数
mapred.reduce.slowstart.completed.maps配置reduce任务在map任务完成到百分之几的时候开始进入
这个几个参数要看实际节点的情况进行配置，reduce任务是在33%的时候完成copy，要在这之前完成map任务，（map可以提前完成）
mapred.compress.map.output,mapred.output.compress配置压缩项，消耗cpu提升网络和磁盘io
合理利用combiner
注意重用writable对象

232. Hadoop高并发?

首先肯定要保证集群的高可靠性，在高并发的情况下不会挂掉，支撑不住可以通过横向扩展。
datanode挂掉了使用hadoop脚本重新启动。

233. hadoop的TextInputFormat作用是什么，如何自定义实现?

InputFormat会在map操作之前对数据进行两方面的预处理。
1是getSplits，返回的是InputSplit数组，对数据进行split分片，每片交给map操作一次 。
2是getRecordReader，返回的是RecordReader对象，对每个split分片进行转换为key-value键值对格式传递给map。
常用的InputFormat是TextInputFormat，使用的是LineRecordReader对每个分片进行键值对的转换，以行偏移量作为键，行内容作为值。
自定义类继承InputFormat接口，重写createRecordReader和isSplitable方法 。
在createRecordReader中可以自定义分隔符。

234. hadoop和spark的都是并行计算，那么他们有什么相同和区别?

两者都是用mr模型来进行并行计算，hadoop的一个作业称为job，job里面分为map task和reducetask，每个task都是在自己的进程中运行的，当task结束
时，进程也会结束。
spark用户提交的任务成为application，一个application对应一个sparkcontext，app中存在多个job，每触发一次action操作就会产生一个job。
这些job可以并行或串行执行，每个job中有多个stage，stage是shuffle过程中DAGSchaduler通过RDD之间的依赖关系划分job而来的，每个stage里面有多
个task，组成taskset有TaskSchaduler分发到各个executor中执行，executor的生命周期是和app一样的，即使没有job运行也是存在的，所以task可以快速启
动读取内存进行计算。
hadoop的job只有map和reduce操作，表达能力比较欠缺而且在mr过程中会重复的读写hdfs，造成大量的io操作，多个job需要自己管理关系。
spark的迭代计算都是在内存中进行的，API中提供了大量的RDD操作如join，groupby等，而且通过DAG图可以实现良好的容错。

235. 为什么要用flume导入hdfs，hdfs的构架是怎样的?

flume可以实时的导入数据到hdfs中，当hdfs上的文件达到一个指定大小的时候会形成一个文件，或者超过指定时间的话也形成一个文件。
文件都是存储在datanode上面的，namenode记录着datanode的元数据信息，而namenode的元数据信息是存在内存中的，所以当文件切片很小或者很多
的时候会卡死。

236. map-reduce程序运行的时候会有什么比较常见的问题?

比如说作业中大部分都完成了，但是总有几个reduce一直在运行。
这是因为这几个reduce中的处理的数据要远远大于其他的reduce，可能是因为对键值对任务划分的不均匀造成的数据倾斜。
解决的方法可以在分区的时候重新定义分区规则对于value数据很多的key可以进行拆分、均匀打散等处理，或者是在map端的combiner中进行数据预处
理的操作。

237. 简单说一下hadoop和spark的shuffle过程?

hadoop：map端保存分片数据，通过网络收集到reduce端。
spark：spark的shuffle是在DAGSchedular划分Stage的时候产生的，TaskSchedule要分发Stage到各个worker的executor。
减少shuffle可以提高性能。

238. RDD机制?

rdd分布式弹性数据集，简单的理解成一种数据结构，是spark框架上的通用货币。
所有算子都是基于rdd来执行的，不同的场景会有不同的rdd实现类，但是都可以进行互相转换。
rdd执行过程中会形成dag图，然后形成lineage保证容错性等。
从物理的角度来看rdd存储的是block和node之间的映射。

239. spark有哪些组件?

（1）master：管理集群和节点，不参与计算。
（2）worker：计算节点，进程本身不参与计算，和master汇报。
（3）Driver：运行程序的main方法，创建spark context对象。
（4）spark context：控制整个application的生命周期，包括dagsheduler和task scheduler等组件。
（5）client：用户提交程序的入口。

240. spark工作机制?

用户在client端提交作业后，会由Driver运行main方法并创建spark context上下文。
执行add算子，形成dag图输入dagscheduler，按照add之间的依赖关系划分stage输入task scheduler。
task scheduler会将stage划分为task set分发到各个节点的executor中执行。

241. spark的优化怎么做?

通过spark-env文件、程序中sparkconf和set property设置。
（1）计算量大，形成的lineage过大应该给已经缓存了的rdd添加checkpoint，以减少容错带来的开销。
（2）小分区合并，过小的分区造成过多的切换任务开销，使用repartition。

242. kafka工作原理?

producer向broker发送事件，consumer从broker消费事件。
事件由topic区分开，每个consumer都会属于一个group。
相同group中的consumer不能重复消费事件，而同一事件将会发送给每个不同group的consumer。

243. ALS算法原理?

答：对于user-product-rating数据，als会建立一个稀疏的评分矩阵，其目的就是通过一定的规则填满这个稀疏矩阵。
als会对稀疏矩阵进行分解，分为用户-特征值，产品-特征值，一个用户对一个产品的评分可以由这两个矩阵相乘得到。
通过固定一个未知的特征值，计算另外一个特征值，然后交替反复进行最小二乘法，直至差平方和最小，即可得想要的矩阵。

244. kmeans算法原理?

随机初始化中心点范围，计算各个类别的平均值得到新的中心点。
重新计算各个点到中心值的距离划分，再次计算平均值得到新的中心点，直至各个类别数据平均值无变化。

245. canopy算法原理?

根据两个阈值来划分数据，以随机的一个数据点作为canopy中心。
计算其他数据点到其的距离，划入t1、t2中，划入t2的从数据集中删除，划入t1的其他数据点继续计算，直至数据集中无数据。

246. 朴素贝叶斯分类算法原理?

对于待分类的数据和分类项，根据待分类数据的各个特征属性，出现在各个分类项中的概率判断该数据是属于哪个类别的。

247. 关联规则挖掘算法apriori原理?

一个频繁项集的子集也是频繁项集，针对数据得出每个产品的支持数列表，过滤支持数小于预设值的项，对剩下的项进行全排列，重新计算支持
数，再次过滤，重复至全排列结束，可得到频繁项和对应的支持数。

248. 简答说一下hadoop的map-reduce编程模型

首先map task会从本地文件系统读取数据，转换成key-value形式的键值对集合
使用的是hadoop内置的数据类型，比如longwritable、text等
将键值对集合输入mapper进行业务处理过程，将其转换成需要的key-value在输出
之后会进行一个partition分区操作，默认使用的是hashpartitioner，可以通过重写hashpartitioner的getpartition方法来自定义分区规则
之后会对key进行进行sort排序，grouping分组操作将相同key的value合并分组输出，在这里可以使用自定义的数据类型，重写WritableComparator的
Comparator方法来自定义排序规则，重写RawComparator的compara方法来自定义分组规则
之后进行一个combiner归约操作，其实就是一个本地段的reduce预处理，以减小后面shufle和reducer的工作量
reducetask会通过网络将各个数据收集进行reduce处理，最后将数据保存或者显示，结束整个job

249. hadoop的TextInputFormat作用是什么，如何自定义实现

InputFormat会在map操作之前对数据进行两方面的预处理
1是getSplits，返回的是InputSplit数组，对数据进行split分片，每片交给map操作一次
2是getRecordReader，返回的是RecordReader对象，对每个split分片进行转换为key-value键值对格式传递给map
常用的InputFormat是TextInputFormat，使用的是LineRecordReader对每个分片进行键值对的转换，以行偏移量作为键，行内容作为值
自定义类继承InputFormat接口，重写createRecordReader和isSplitable方法
在createRecordReader中可以自定义分隔符

250. hadoop和spark的都是并行计算，那么他们有什么相同和区别

两者都是用mr模型来进行并行计算，hadoop的一个作业称为job，job里面分为map task和reducetask，每个task都是在自己的进程中运行的，当task结束
时，进程也会结束
spark用户提交的任务成为application，一个application对应一个sparkcontext，app中存在多个job，每触发一次action操作就会产生一个job
这些job可以并行或串行执行，每个job中有多个stage，stage是shuffle过程中DAGSchaduler通过RDD之间的依赖关系划分job而来的，每个stage里面有多
个task，组成taskset有TaskSchaduler分发到各个executor中执行，executor的生命周期是和app一样的，即使没有job运行也是存在的，所以task可以快速启
动读取内存进行计算
hadoop的job只有map和reduce操作，表达能力比较欠缺而且在mr过程中会重复的读写hdfs，造成大量的io操作，多个job需要自己管理关系
spark的迭代计算都是在内存中进行的，API中提供了大量的RDD操作如join，groupby等，而且通过DAG图可以实现良好的容错

251. 为什么要用flume导入hdfs，hdfs的构架是怎样的

flume可以实时的导入数据到hdfs中，当hdfs上的文件达到一个指定大小的时候会形成一个文件，或者超过指定时间的话也形成一个文件
文件都是存储在datanode上面的，namenode记录着datanode的元数据信息，而namenode的元数据信息是存在内存中的，所以当文件切片很小或者很多
的时候会卡死

252. map-reduce程序运行的时候会有什么比较常见的问题

比如说作业中大部分都完成了，但是总有几个reduce一直在运行
这是因为这几个reduce中的处理的数据要远远大于其他的reduce，可能是因为对键值对任务划分的不均匀造成的数据倾斜
解决的方法可以在分区的时候重新定义分区规则对于value数据很多的key可以进行拆分、均匀打散等处理，或者是在map端的combiner中进行数据预处
理的操作

253. 简单说一下hadoop和spark的shuffle过程

hadoop：map端保存分片数据，通过网络收集到reduce端
spark：spark的shuffle是在DAGSchedular划分Stage的时候产生的，TaskSchedule要分发Stage到各个worker的executor
减少shuffle可以提高性能

254. 谈谈RDD、DataFrame、Dataset的区别和各自的优势?

共性：
1、RDD、DataFrame、Dataset全都是spark平台下的分布式弹性数据集，为处理超大型数据提供便利。
2、三者都有惰性机制，在进行创建、转换，如map方法时，不会立即执行，只有在遇到Action如foreach时，三者才会开始遍历运算。
3、三者都会根据spark的内存情况自动缓存运算，这样即使数据量很大，也不用担心会内存溢出。
4、三者都有partition的概念。
5、三者有许多共同的函数，如filter，排序等
6、在对DataFrame和Dataset进行操作许多操作都需要这个包进行支持：
import spark.implicits._
7、DataFrame和Dataset均可使用模式匹配获取各个字段的值和类型。
区别：
RDD:
1、RDD一般和spark mlib同时使用
2、RDD不支持sparksql操作
DataFrame:
1、与RDD和Dataset不同，DataFrame每一行的类型固定为Row，只有通过解析才能获取各个字段的值。
2、DataFrame与Dataset均支持sparksql的操作，比如select，groupby之类，还能注册临时表/视窗，进行sql语句操作。
3、DataFrame与Dataset支持一些特别方便的保存方式，比如保存成csv，可以带上表头，这样每一列的字段名一目了然。
Dataset:
这里主要对比Dataset和DataFrame，因为Dataset和DataFrame拥有完全相同的成员函数，区别只是每一行的数据类型不同。
DataFrame也可以叫Dataset[Row],每一行的类型是Row，不解析，每一行究竟有哪些字段，各个字段又是什么类型都无从得知，只能用上面提到的getAS方法或者共性中的第七条提到的模式匹配拿出特定字段。
而Dataset中，每一行是什么类型是不一定的，在自定义了case class之后可以很自由的获得每一行的信息。
2、Spark技术栈有哪些组件，每个组件都有什么功能，适合什么应用场景？
1）Spark core：是其它组件的基础，spark的内核，主要包含：有向循环图、RDD、Lingage、Cache、broadcast等，并封装了底层通讯框架，是Spark的基础。
2）SparkStreaming是一个对实时数据流进行高通量、容错处理的流式处理系统，可以对多种数据源（如Kafka、Flume、Twitter、Zero和TCP 套接字）进行类似Map、Reduce和Join等复杂操作，将流式计算分解成一系列短小的批处理作业。
3）Spark sql：Shark是SparkSQL的前身，Spark SQL的一个重要特点是其能够统一处理关系表和RDD，使得开发人员可以轻松地使用SQL命令进行外部查询，同时进行更复杂的数据分析。
4）BlinkDB ： 是一个用于在海量数据上运行交互式 SQL 查询的大规模并行查询引擎，它允许用户通过权衡数据精度来提升查询响应时间，其数据的精度被控制在允许的误差范围内。
5）MLBase是Spark生态圈的一部分专注于机器学习，让机器学习的门槛更低，让一些可能并不了解机器学习的用户也能方便地使用MLbase。MLBase分为四部分：MLlib、MLI、ML Optimizer和MLRuntime。
6）GraphX是Spark中用于图和图并行计算

255. RDD机制?

rdd分布式弹性数据集，简单的理解成一种数据结构，是spark框架上的通用货币。
所有算子都是基于rdd来执行的，不同的场景会有不同的rdd实现类，但是都可以进行互相转换。
rdd执行过程中会形成dag图，然后形成lineage保证容错性等。
从物理的角度来看rdd存储的是block和node之间的映射。

256. spark有哪些组件?

（1）master：管理集群和节点，不参与计算。
（2）worker：计算节点，进程本身不参与计算，和master汇报。
（3）Driver：运行程序的main方法，创建spark context对象。
（4）spark context：控制整个application的生命周期，包括dagsheduler和task scheduler等组件。
（5）client：用户提交程序的入口。

257. spark工作机制?

用户在client端提交作业后，会由Driver运行main方法并创建spark context上下文。
执行add算子，形成dag图输入dagscheduler，按照add之间的依赖关系划分stage输入task scheduler。
task scheduler会将stage划分为task set分发到各个节点的executor中执行。

258. spark的优化怎么做?

通过spark-env文件、程序中sparkconf和set property设置。
（1）计算量大，形成的lineage过大应该给已经缓存了的rdd添加checkpoint，以减少容错带来的开销。
（2）小分区合并，过小的分区造成过多的切换任务开销，使用repartition。

259. 什么是YARN模式?

在YARN模式下，YARN ResourceManager执行Spark Master的功能。 Workers的功能由运行执行程序的YARN NodeManager守护程序执行。 YARN模式设置稍微复杂一些，但它支持安全性。

260. 什么是客户端模式和集群模式?

每个应用程序都有一个协调其执行的驱动程序进程 此过程可以在前台（客户端模式）或后台（群集模式）中运行。 客户端模式稍微简单一些，但集群模式允许在启动Spark应用程序后轻松注销，而无需终止应用程序。

261. 如何在独立客户端模式(client mode)下运行spark?

```
spark-submit \
class org.apache.spark.examples.SparkPi \
deploy-mode client \
master spark//$SPARK_MASTER_IP:$SPARK_MASTER_PORT \
$SPARK_HOME/examples/lib/spark-examples_version.jar 10
```

262. 如何在独立群集模式(cluster mode)下运行spark?

```
spark-submit \
class org.apache.spark.examples.SparkPi \
deploy-mode cluster \
master spark//$SPARK_MASTER_IP:$SPARK_MASTER_PORT \
$SPARK_HOME/examples/lib/spark-examples_version.jar 10
```

263. 如何在YARN客户端模式下（client mode）运行spark?

```
spark-submit \
class org.apache.spark.examples.SparkPi \
deploy-mode client \
master yarn \
$SPARK_HOME/examples/lib/spark-examples_version.jar 10
```

264. 如何在YARN集群模式（cluster mode）下运行spark?

```
spark-submit \
class org.apache.spark.examples.SparkPi \
deploy-mode cluster \
master yarn \
$SPARK_HOME/examples/lib/spark-examples_version.jar 10
```

265. 什么是Executor内存?

可以使用sparksubmit的-executor-memory参数来配置它。 默认情况下，此设置为1 GB.

266. 定义RDD?

RDD（弹性分布式数据集）是Apache Spark中的基本抽象，表示以对象格式进入系统的数据。 RDD以容错方式用于大型集群上的内存计算。 RDD是只读的分区，记录集合，即 -
不可变 - RDD不能改变。
弹性 - 如果持有分区的节点失败，则另一个节点获取数据。

267. Spark对MapReduce优势?

Spark与MapReduce相比具有以下优势：

- 由于内存处理的可用性，Spark实现的处理速度比Hadoop MapReduce快10到100倍，而MapReduce则使用持久性存储来执行任何数据处理任务。
- 与Hadoop不同，Spark提供了内置库，可以从同一个核心执行多个任务，如批处理，Steaming，机器学习，Interactive SQL查询。 但是，Hadoop仅支持批处理。
- Hadoop与磁盘高度相关，而Spark则提升缓存和内存数据存储。
- Spark能够在同一数据集上多次执行计算。 这称为迭代计算，而Hadoop没有直接实现迭代计算。